XSS attack detection using machine learning

jayshree.karmakar2020@vitbhopal.ac.in

July 2022

1 Introduction

2 Related works

Since around 68 percent of websites were susceptible to XSS attacks, Habibi et al.[8] devised a strategy to stop them using machine learning detection techniques. The authors improved the detection performance of XSS attacks by using the n-gram method to script each feature and the machine learning techniques SVM, Naïve Bayes, and KNN to determine the best classifier.According to them, the SVM and n-gram approach had a 98 percent accuracy rate. The machine learning detection techniques, which had been trained to recognize malicious scripts frequently used in XSS attacks, was one approach to prevent these XSS attack.In this research, Prasetio et al. [17] discussed the classification accuracy of cross-site scripting (XSS) attacks by combining two techniques for identifying feature characteristics, namely feature selection and language computation. With an accuracy value of 99. 87 percent and a low false positive rate of 0.039 percent,they deduced from the data that hybrid feature modelling provided them a good accuracy. This made this research study distinct from others. Logistic regression was used in this study because its known for the fast processing and good accuracy.To combat XSS attacks, Tariq et al. [18] suggested using genetic algorithms (GA), reinforcement learned (RL), and threat intelligence. The suggested approach not only employed a white-box model but also provided the end users with clear results.The suggested solution was sound and had successfully decreased false positives. Previous methods were flawed because they couldn't deal with the problem of endless input space. This suggested method addressed those by using patterns . The study's findings demonstrated that the suggested strategy on the Deurgan dataset achieved an accuracy of 99. 75 percent in the default setting and up to 99. 89 percent in its worst case (after injecting malicious payloads). Attacks using cross-site scripting (XSS) are among the attacks on the web. As a countermeasure of XSS attack, machine learning had attracted a lot of attention. In the previous researches, there were problems that the size of data set was too small or unbalanced, and that prepossessing method for vectorization of strings caused misclassification.The authors Aksishi et al. [2] of this study used word2vec that

determined the frequency of appearance and co-occurrence of the words in the XSS attacked scripts, which improved the prepossessing procedure for vectorization.To reduce the data's variance, they also used a sizable data collection. Finally, in the end, they demonstrated that their approach was effective in a practical setting."In this study, Wang et al. [19] proposed an adversarial attack model based on soft q-learning,which could generate adversarial multi-fold strategies. They constructed a fuzz data set containing 243,621 strings. They then demonstrated how their proposed approach could be used to generate adversarial attacks against different XSS attacked detection models.

In this paper,Hoang et al.[9] discussed common web exploits like SQL injection, cross-site scripting, operating system command injection, and path traversal in their article. They suggested a low-cost detection model created with a decision tree method that was based on machine learning using weblogs. The detection model was built using a machine learning technique, which replaced manual construction and updated input data filters that improved the detection rates.The detection model also employed weblogs produced by web servers for each hosted website as input.The model had an overall detection accuracy rate of 98. 56 percent and could efficiently detect web attacks.In this paper, the authors Kaur et al. [14] studied and analyzed blind XSS attacks and their detection by using machine learning techniques. They were interested to analyze a new dataset to extract various features of payloads that could be used as a benchmark for the OWASP community. Blind XSS attacks could be detected with 95. 4 percent of accuracy with their above-proposed method.

Kascheev et al. [13] provided an explanation of XSS attacks in this study and suggested a machine learning technique to identify them. They attempted to contrast and evaluate the various algorithms in this paper. These machine learning techniques were contrasted for accuracy. For XSS detection using ML, they have employed binarization of inbound requests. Before the training, they had shown each inquiry as a vector of token words. After that, they created training and test samples using cross-validation. The researchers employed supervised learning methods.The decision tree model was regarded as the best model based on the investigators' findings. In this paper, Ivanova et al.[11] established a machine learning technique for the detection of stored XSS attacks and the security of REST web services. For this, a JAVA-based REST web service was developed. The simulation, which comprised Postman, IntelliJ IDEA, and an internet browser, was used to study and attack it after that. The clustering algorithm k-means was employed to identify unknown threats, and the supervised machine learning classifiers were utilised to detect known attacks. The analysis of the algorithms revealed that they were very precise. There was also application of fuzzy sets and fuzzy logic theory. In this study, Alquarni et al.[3] applied a novel deep learning algorithm called MNN to reduce the false positive rate of an XSS attack detection system.The results of the experiments indicated that the suggested method learned to detect malicious JavaScript-based XSS attacks with an accuracy of 99. 96 percent. The feature selection process heavily relied on the Pearson correlation, which increased the accuracy.

3 Methodology

3.1 Background

3.1.1 Definition

Vicious scripts are fitted into else secure and innocent websites in Cross-Site Scripting (XSS) attacks. XSS attacks take place when a bushwhacker sends vicious code, generally in the form of a cybersurfed side script, to a separate end stoner using an online operation. The ubiquitous flaws that enable these attacks to succeed happen whenever a web application incorporates user input into the output it produces without verifying or encoding it. A vicious script can be transferred to an unwary stoner by a bushwhacker using XSS. The end stoner's cybersurfer will run the script anyhow of the fact that it should not be trusted. The vicious script is suitable to pierce any eyefuls, session tokens, or other sensitive data stored by the cybersurfer and used with that point since it believes the script is from a dependable source. These scripts have the capability to fully change HTML runner content.

3.1.2 Types Of XSS Attacks

Stored XSS: Attackers who use stored XSS insert malicious scripts into web applications that are subsequently stored on the target server as database records or web server logs. The malicious script is then retrieved from the website by a good user. Once the malicious script has access to the user's session cookies, it can operate on the user's behalf.

Reflected XSS: The most prevalent type of XSS is reflected XSS, which is also the easiest for attackers to execute. The attacker invites the user to click on the URL that contains malicious code through social engineering. The intrusion is made easier by the acquisition of user cookies, which can be used to hijack user sessions. In a reflected XSS attack, the web application returns the data right away without first securing it for the browser.

DOM-based XSS: A sub type of both persistent and reflected XSS is DOM-based XSS. The malicious string is not actually processed by the victim's browser during a DOM-based XSS attack until the legal JavaScript for the website is run. The attack payload and the full data flow between source and sink in a DOM-based XSS exploit never leave the browser.

Blind XSS: A kind of persistent XSS vulnerabilities is blind XSS vulnerabilities. They take place when the server saves the attacker's input and displays it in another area of the programme or in another application. For instance, if an attacker inserts a malicious payload into a contact or feedback page, the attacker's payload will be loaded when the programme administrator is examining the feedback entries. Execution of the attacker's input is possible in a different application .

3.1.3 Working Of XSS attacks

Cross-scripting is a type of web application hacking that only requires a victim, a website, and a web server. All you need to view a website is a laptop and a strong internet connection. When you use the web application, data exchange between you and the web server occurs via this website i.e. when you send a request to the web server, the web server responds by sending you a response via a web page on the website. Depending on the type of XSS being used, a hacker can inject malicious code on a website, which is then sent either to the victim or the web-server. The malicious script is either executed when the victim visits the website or when the victim attempts to access a page or some data from the web-server. A hacker can then inject such code that can be used to steal the credentials or any sensitive information.

Attacks using cross-site scripting could result in the following:

- Monitoring a user's keystrokes.
- Sending a visitor to a harmful website.
- Utilising web browser exploits (e.g., crashing the browser).
- Obtaining a website user's cookie information and using it to compromise the victim's account.

In some instances, an XSS attack causes the victim's account to be totally compromised. Users can be duped into entering their credentials on a false form, which gives the attacker access to all the data.

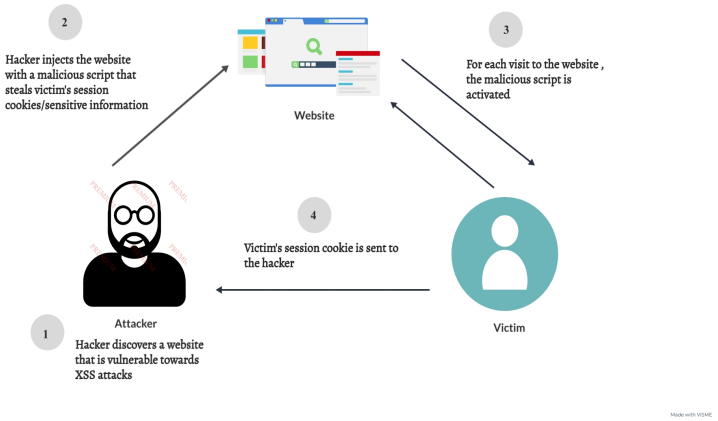

fig 1: How XSS-attack works

3.1.4 Diagrams

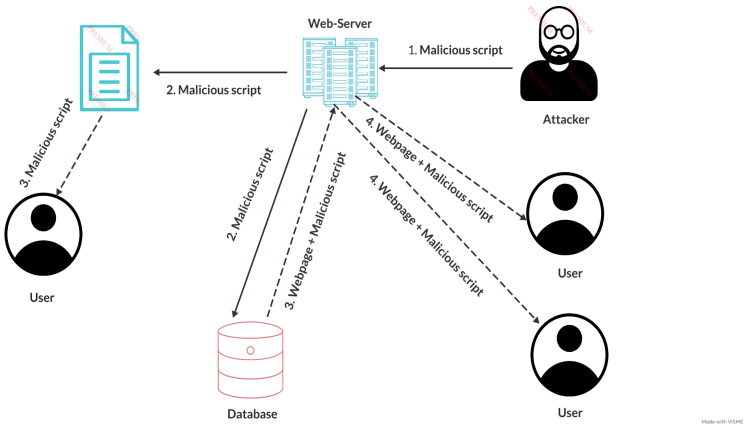

fig 2: Stored XSS attack data flow

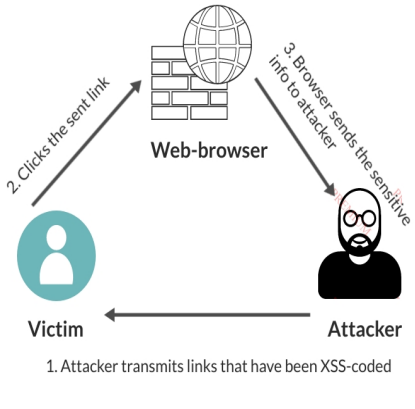

fig 3: Reflected XSS attack data flow

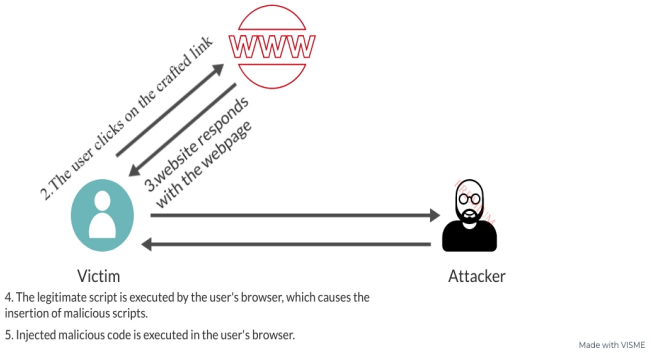

fig 4: Dom-Based XSS attack data flow

3.1.5 Software or tools available to detect XSS attacks

XSS discloses your application to numerous security flaws that may harm businesses and public consider an XSS attack that compromises a customer's bank account on a bank website this could have a negative impact on the banks reputation and image consequently there are many various tools and software available to analyse XSS vulnerabilities in order to stop them.

- In XSS detection and prevention approach, WAFs (Web Application Firewall) are used.
- Developers use filtering input and output encoding to stop XSS injection attacks.

To find XSS issues, certain automated application testing techniques are available.

1. SAST (Static Application Security Testing) employs a white-box testing approach to examine the application's source code and its constituent parts for potential security issues.

2. DAST (Dynamic Application Security Testing) employs a black-box testing strategy. In a dynamic test, fuzzing involves feeding the application with erroneous, corrupted data to see if it will detect XSS flaws.

3. SAST and DAST are combined into IAST (Interactive Application Security Testing), while the programme is running, examines the code for XSS by actively utilising its features.

4. Web scanners are effective programmes that can crawl your entire website or application and automatically check for XSS and other security problems. Examples include Invicti, Acunetix, Veracode, and Checkmarx.

3.1.6 Role of ML in detection of XSS attacks

The previous methods to detect XSS loopholes can easily escape the firewall via attack payloads. Machine learning techniques can be used to detect the XSS vulnerabilities by collecting data sets(attack payloads) from different sources , test and train those models with various existing ML algorithms to get better accuracy and performance.

3.2 Problem statement

Web attacks are continuous increasingly even though many protocols and standards provide web security. According to OWASP in 2013, XSS was ranked A3 in top 10 and in 2017, XSS was ranked A7 in top 10. XSS remains a top vulnerability out of 10 [1]. According to OWSAP injection attacks dropped to the 3rd position. 94 applications underwent testing for injection of some kind. (Max. incidence 19 percent avg. incidence rate 3.37 percent) [15]. According to the latest published CVE report (i.e., NOV 21 to JAN 22) XSS remains first ranked. XSS attacks are mostly classified to have medium to high severity [6]. The largest increase in XSS attacks since Q4 2015 occurred in Q1 of 2017[16].80 percent of cybersecurity sectors are moving towards adaption of ML as per .ML And NN are recently been used to detect XSS loopholes in the source code.ML models are been trained and tested on different data-sets to improve the accuracy and performance. Attackers can easily bypass the existing security standards, mitigation techniques, filters, encoders, intrusion detection, firewalls and prevention methods by injecting new attack payloads. Let's say that an algorithm has been trained to recognise payloads that contain alert tags and a script. The taught algorithm can therefore be easily tricked by malicious attack payloads using newer functions and patterns.

Many Researches tried to improve the detection performance of XSS attacks by using different methods to each script feature and the machine learning techniques SVM, Naive Bayes, and KNN to determine the best classifier by combining two techniques or multiple models for identifying feature characteristics [1,4,5,6,8]. To build low-cost detection and affordable model though predominantly available, only a handful of researchers [3,9] have touched upon the issue. Some researchers tried to test on real-environment, extensive testing hardware which increased the time, cost and effort. The training approach becomes ineffective when the variety of new injection payloads and feature sets evolve, which increases the computational complexity.

In our approach,I have used the AdaBoost algorithm for classifying web-based app inputs as XSS attacks or benign inputs.I will try to build a best optimal model to better the prediction and reduce time and effort.To improve the pre-processing for vectorization I tried to use count vectorization to extract features from text XSS attack scripts, I tried to employ a substantial data collection to lower the variance of the data. The approach will adapt to the new attack environment with a recent feature set with less effort after learning differences in the recent malicious payloads moreover this approach is carried

out on recent dataset.

4 Implementation

4.1 Architecture

Our work is to give a model for detecting XSS attacks. The data-set which is collected is then analysed using ML-based approach. If the intercepted requests do not have the features of adversarial attack payloads, then the requests are dropped and the application gets protected from XSS which the basic working of any XSS detector that is built using ML approach. Features are used to refer to the most common JavaScript arguments. Binarization of incoming requests is the challenge of identifying cross-site scripting using machine learning techniques. Incoming requests could be related to either attack or benign code. Each query must be presented as a vector of token words before training. In JavaScript, these tokens are request parameters.

Cross-validation is a technique where the original data-set is divided into training and testing samples. By employing the complementary subset of the data set to test our model after it has been trained using the subset of the original data set this process takes place. When evaluating the algorithm's output using metrics or limited sample data, the test sample is employed. The generated model is then checked for accuracy rate, precision, f1-score.

Since detection of XSS is a binary classification task therefore I have used AdaBoost algorithm since it is a boosting classifier and it can be combined with both weak base and strong base learner to improve accuracy.

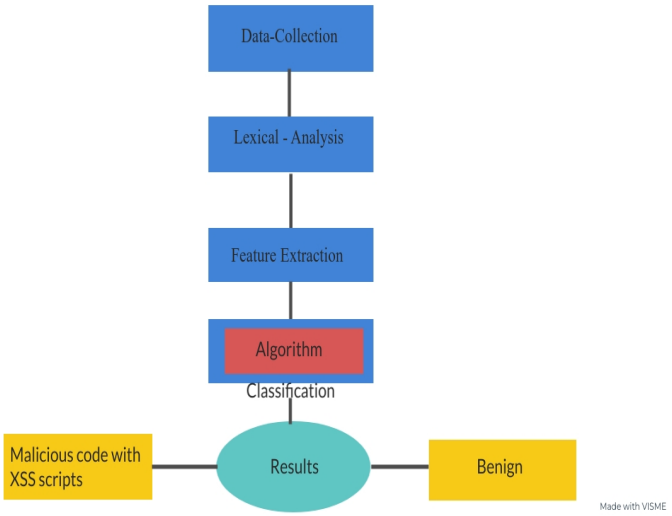

fig 5: Overall System Architecture

A. Data Collection

To train a model in ML with a good accuracy we need to collect and prepare the source dataset correctly. Since it is a detection model therefore, we need to collect data of both harmful and benign JavaScript code. For the purpose of identifying XSS attacks, I have gathered recent data from the portswigger cheatsheet and the OWSAP cheatsheet. Label columns have target labels, and sentence contains script data. The dataset consists of 6300 examples of malicious code and 7300 examples of benign code.

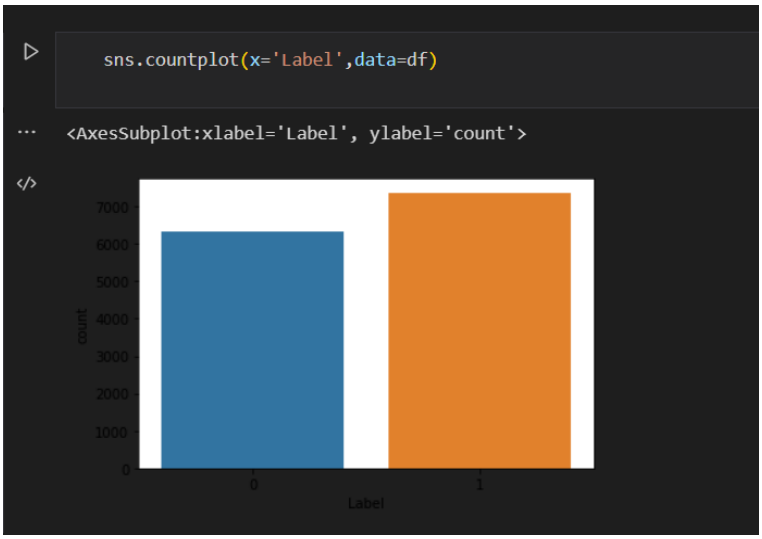

fig 6: Count for malicious and benign code

B. Preprocessing and Feature extraction

The first information that is gathered is then divided into test and training set with a 20 to 80 proportion. The ML model that is planned will be utilized to train the information and the test informational index will be utilized to assess the model's presentation. The contribution of the AI calculation is the changed rendition of preparing and test informational index. The source informational collection is addressed as queries with an alternate number of boundaries. Each solicitation is decoded into Unicode characters. In the wake of unravelling, boundaries of this query are extricated utilizing standard articulations. Unusual queries containing countless boundaries are taken out from the dataset. Tokenizing text with CountVectorizer entails breaking down sentences into words and conducting very minimal pre-processing. The punctuation is removed, and all the words are changed to lowercase.In this case, binary was employed, which is typically done when the machine learning model cannot learn anything helpful from the phrase or word count.By setting "binary = True," the CountVectorizer stops taking the term's or word's frequency into account. It is set to 1 if it occurs, 0 otherwise. Binary is set to False by default.

```
text = ''.join(df['Sentence'].tolist())

word_cloud = WordCloud(collocations = False, background_color = 'white').generate(text)

plt.imshow(word_cloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

fig 7: pre-processing

C. Classification

Since the raw data set contains both benign and attack samples therefore, we require binary classification. Binarization is the task of segregating positive attack sample from negative attack sample. The classification algorithms that are frequently used are Logistic Regression, Naïve Bayes, Support Vector Machines (SVM), Random Forest, Neural Networks, Bagging and Boosting, Deep Learning. In my approach I have used AdaBoost algorithm since it is a boosting classifier and it can be combined with both weak base and strong base learner to improve accuracy.

4.2 Implementation detailing

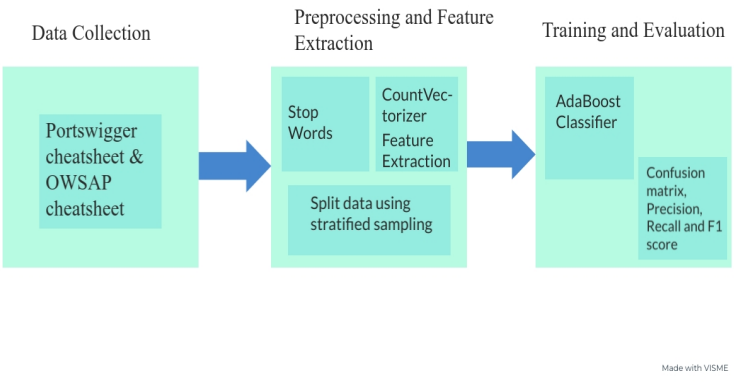

fig 6: Machine Learning Pipeline

A. Hardware and Software Requirements

We employed the Python programming language and its libraries for this research. Python was selected because it includes a library for doing sophisticated mathematical operations. The best language in which machine learning can be done is python because it's easy to understand. There are many external libraries available for deep learning, data processing etc in python. The platform on which the codes were written is Kaggle workspace. The research is done on windows10 but any platform can be used like Linux or mac.

B. Algorithms

We have employed the AdaBoost algorithm in our strategy to categorise inputs as XSS attacks or benign inputs. Yoav Freund and Robert Schapire developed the statistical classification meta-algorithm known as AdaBoost in 1995. We picked AdaBoost because it is used for binary classification since that is what our challenge required.AdaBoost algorithm for binary classification task.The below definition is referred from [7].

Initialization:

1. Given training data from the instance space $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} = \{-1, +1\}$.
2. Initialize the distribution $D_1(i) = \frac{1}{m}$.

Algorithm:

for $t = 1, ..., T$: **do**

Train a weak learner $h_t : \mathcal{X} \rightarrow$ R using distribution D_t.

Determine weight α_t of h_t.

Update the distribution over the training set:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

where Z_t is a normalization factor chosen so that D_{t+1} will be a distribution.

end for

Final score:

$f(x) = \sum_{t=0}^{T} \alpha_t h_t(x)$ and $H(x) = sign(f(x))$

C. Training

The hyperparameters of the algorithm determine the classifiers' performance and accuracy. The learning process is controlled by hyperparameters. Hyperparameters must be set by the ML algorithm's user because they cannot be obtained through training. Hyperparameter optimization is the process of identifying a set of ideal hyperparameters for a specific ML method. In our method, we employed grid search to optimise the hyperparameters. We applied k-Fold cross-validation to prevent the algorithm from becoming overfit. A statistical re sampling method called k-fold cross-validation trains and tests the machine learning model on k subsets of training data. In our strategy, we used a cross-validation method with k set to 5. 14 candidates were suited with 5 folds, for a total of 70 fits.The code is given below.

```
# For Adaboost
parameters = {
    'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 20, 30]
}
clf = GridSearchCV(lr_clf, parameters, cv=5, verbose=1, n_jobs=10)
clf.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 14 candidates, totalling 70 fits

GridSearchCV(cv=5, estimator=AdaBoostClassifier(n_estimators=100), n_jobs=10,
             param_grid={'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                          20, 30]},
             verbose=1)
```

5 Results

The collected data is divided into test set and train set using 5- fold cross–validation. The percentages of train and test sample is 80 and 20 percentage respectively. Cross-validation [13] is a procedure for empirically evaluating the generalizing ability of algorithms. Cross-validation emulates the presence of a test sample that does not participate in training, but for which the correct answers are known. Each instance is represented as a token vector. Tokenizing text with Count Vectorizer entails breaking down sentences into words and conducting very minimal pre-processing. Abnormal entries were deleted during this process. Such data may reduce the accuracy of the built model. The process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words[5]. We have used this in our experimental setup to remove the useless data.The dataset is then trained on AdaBoost machine learning model for performance analysis.The platform used was Kaggle workspace and windows10.In our method, we employed grid search to optimise the hyperparameters.The hyperparameters of the algorithm determine the classifiers' performance and accuracy. Metrics is one of the key indicators. It is used to evaluate the result of the algorithm. To evaluate the accuracy of the implemented model we used metrics such as: accuracy, precision, recall and F-measure, specificity and sensitivity. Accuracy – Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations[12].
Accuracy = TP+TN/TP+FP+FN+TN
where: TP (True Positives)-correctly classified positive examples;
TN (True Negatives)-correctly classified negative examples;

FN (False Negatives)-positive examples classified as negative (type I error);
FP (False Positives)-negative examples classified as positive (type II error).

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations[12].

Precision = TP/TP+FP
TP (True Positives)-correctly classified positive examples;
FP (False Positives)-negative examples classified as positive (type II error).

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account [12].
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Sensitivity-The term sensitivity was introduced by Yerushalmy in the 1940s as a statistical index of diagnostic accuracy. It is also called the true positive rate, the recall, or probability of detection. It has been defined as the ability of a test to identify correctly all those who have the disease, which is "true-positive"[4].

Sensitivity = TP/TP+FN
TP (True Positives)-correctly classified positive examples;
FN (False Negatives)-positive examples classified as negative (type I error);
Specificity-It is defined as the ability of a test to identify correctly those who do not have the disease, that is, "true-negatives".It is also called as the true negative rate[4].
Specificity = TP/TP+FN
The confusion matrix for the given model is show as below

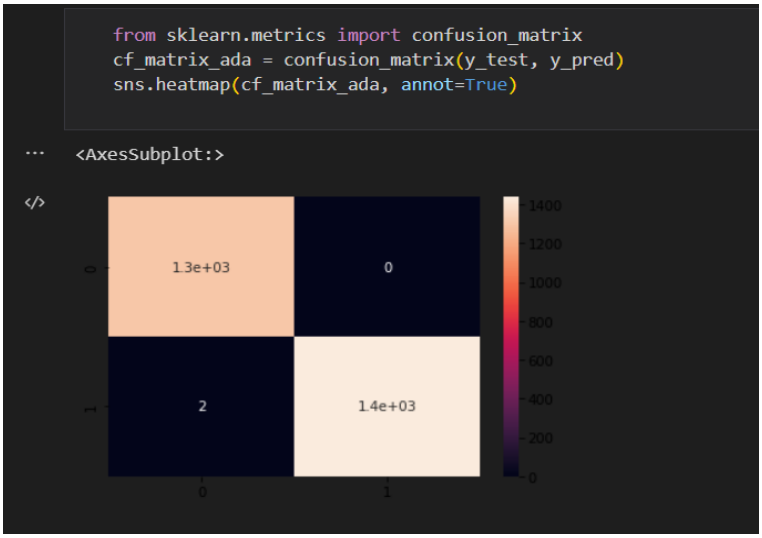

fig 7: Confusion Matrix

The results are shown in the following table :

Metric	Value
Accuracy	0.999
Precision	1.0
F1-score	0.999
Specificity	1.0
Sensitivity	0.998

Table 1: Result-1

Hyperparameters are adjustable parameters that let you control the model training process. For example, with neural networks, you decide the number of hidden layers and the number of nodes in each layer. Model performance depends heavily on hyperparameters. Hyperparameter tuning, also called hyper parameter optimization, is the process of finding the configuration of hyperparameters that results in the best performance. The process is typically computationally expensive and manual[10]. So,I have used hyper-parameter optimization (grid search) in this to obtain best performance of the above model. We have been fitting 5 folds for each of 14 candidates, totalling 70 fits. Confusion matrix for this and the result obtained is given below:

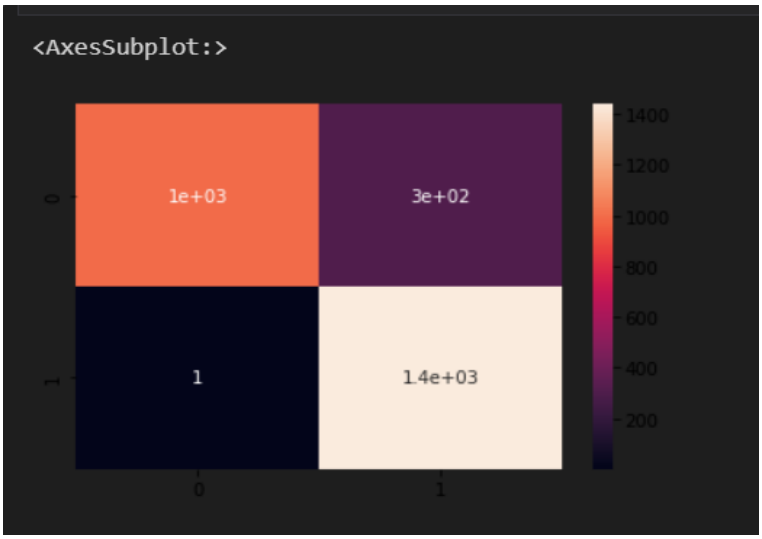

fig 8: Confusion Matrix when applying hyperpameter optimization

The results are shown in the following table :

Metric	Value
Accuracy	0.998
Precision	0.999
F1-score	0.998
Specificity	0.999
Sensitivity	0.998

Table 2: Result-2

There's a slight difference in the scores of precision and specificity when hyper-parameter tuning is applied .The precision and specificity is 1.0 when hyper-parameter optimization isn't applied and when applied it slightly decreases to 0.999.

6 Conclusion and Future work

References

[1] Acunetix. Owasp top 10 2017 update – what you need to know. `https://www.acunetix.com/blog/articles/owasp-top-10-2017/`, 2017.

[2] S. Akaishi and R. Uda. Classification of xss attacks by machine learning with frequency of appearance and co-occurrence. In *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2019.

[3] A. A. Alqarni, N. Alsharif, N. A. Khan, L. Georgieva, E. Pardade, and M. Y. Alzahrani. Mnn-xss: Modular neural network based approach for xss attack detection. *Computers, Materials and Continua*, 70(2):4075–4085, 2022.

[4] S. Aryal. Sensitivity and Specificity- Definition, Formula, Calculation, Relationship. `https://thebiologynotes.com/sensitivity-and-specificity/`, 2022. [Online; accessed 20-Jan-2022].

[5] GeeksForGeeks. Removing stop words with NLTK in Python. `www.geeksforgeeks.org/removing-stop-words-nltk-python/`, 2022. [Online; accessed 22-Aug-2022].

[6] Y. Guan. Vulnerability. `https://unit42.paloaltonetworks.com/network-security-trends-cross-site-scripting/`, May 2022.

[7] U. Guz, S. Cuendet, D. Hakkani-Tur, and G. Tur. Multi-view semi-supervised learning for dialog act segmentation of speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18:320 – 329, 03 2010.

[8] G. Habibi and N. Surantha. Xss attack detection with machine learning and n-gram methods. In *2020 International Conference on Information Management and Technology (ICIMTech)*, pages 516–520. IEEE, 2020.

[9] X. D. Hoang. Detecting common web attacks based on machine learning using web log. In *International Conference on Engineering Research and Applications*, pages 311–318. Springer, 2020.

[10] M. Ignite. Hyperparameter tuning a model (v2). `https://learn.microsoft.com/en-us/azure/machine-learning/how-to-tune-hyperparameters`, 2022. [Online; accessed 12-Oct-2022].

[11] M. Ivanova and A. Rozeva. Detection of xss attack and defense of rest web service–machine learning perspective. In *2021 The 5th International Conference on Machine Learning and Soft Computing*, pages 22–28, 2021.

[12] R. Joshi. How to evaluate the performance of a model in Azure ML and understanding "Confusion Metrics". `https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/`, 2016. [Online; accessed 9-September-2016].

[13] S. Kascheev and T. Olenchikova. The detecting cross-site scripting (xss) using machine learning methods. In *2020 Global Smart Industry Conference (GloSIC)*, pages 265–270. IEEE, 2020.

[14] G. Kaur, Y. Malik, H. Samuel, and F. Jaafar. Detecting blind cross-site scripting attacks using machine learning. In *Proceedings of the 2018 International Conference on Signal Processing and Machine Learning*, pages 22–25, 2018.

[15] OWASP. Welcome to the owasp top 10 - 2021. `https://owasp.org/Top10/`, May 2021.

[16] G. Podjarny. Xss attacks: The next wave. `https://snyk.io/blog/xss-attacks-the-next-wave/`, 2017.

[17] D. A. Prasetio, K. Kusrini, and M. R. Arief. Cross-site scripting attack detection using machine learning with hybrid features. *JURNAL INFOTEL*, 13(1):1–6, 2021.

[18] I. Tariq, M. A. Sindhu, R. A. Abbasi, A. S. Khattak, O. Maqbool, and G. F. Siddiqui. Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning. *Expert Systems with Applications*, 168:114386, 2021.

[19] Q. Wang, H. Yang, G. Wu, K.-K. R. Choo, Z. Zhang, G. Miao, and Y. Ren. Black-box adversarial attacks on xss attack detection model. *Computers & Security*, 113:102554, 2022.