World Journal of Innovation and Modern Technology E-ISSN 2756-5491 P-ISSN 2682-5910
Vol 8. No. 2 2024 www.iiardjournals.org Online Version

Cyber Security Techniques for Detecting and Preventing Cross-Site Scripting Attacks

Okusi, Oluwatobiloba
Bristol Waste Company, Albert Road, Bristol BS2 0XS, UK
Tobi.Okusi@bristolwastecompany.co.uk

DOI: 10.56201/wjimt.v8.no2.2024.pg71.89

Abstract

As the demand as well as use of web application increases daily so also different cyber security threats to it increases alarmingly. Cross-Site Scripting (XSS) attack is one in which an attacker exploits website and web application vulnerabilities by injecting arbitrary web requests into a web page viewed by other users or store unauthorized cookies, thereby attacking and causing harms to the owners' sites and accounts. This study explores XSS attacks, and proposes some cyber security techniques for detecting and preventing XSS. The results of the analysis show that the proposed Deep Forest (DF) model alongside some other AI techniques can help address the issue of class imbalance, an aspect of XSS neglected by extant studies. In conclusion, the study contributes to solving some critical issues of cyber security. Researchers are charged to take up more studies on XSS, with specific focus on DF and class imbalance in cyber security, towards addressing more national and international cyber security threats and effects.

Keywords: *Cyber security, Cross-site scripting, Techniques, Detecting, Preventing*

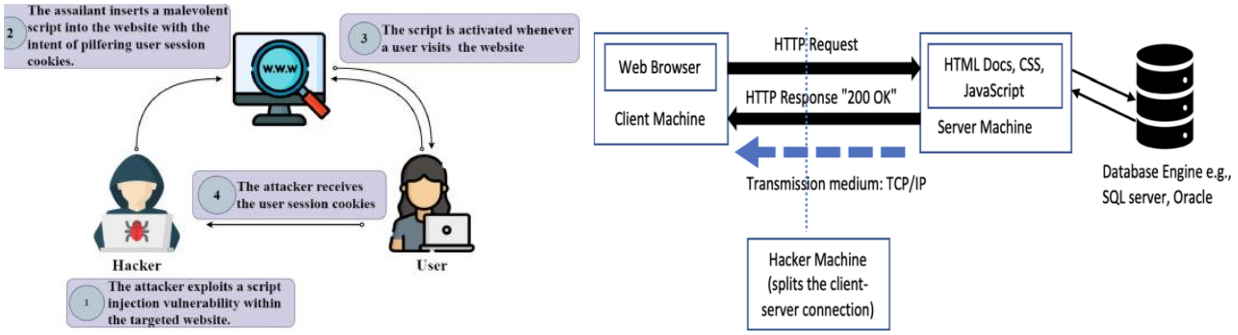
Introduction

The modern information and communication technology (ICT) offers a lot of prospects in all spheres of life. ICT involves as well as relies on the internet. The internet is a pool of computers connected via a network, where seamless communication and data, ideas, views, thoughts, messages, etc. are produced, disseminated, stored and shared among billions of users, though with different sets of users in various settings. These are made possible through a standard protocol known as the Transmission Control Protocol (TCP) or Internet Protocol (IP) that allows for gathering, sharing and storing information on the internet, web inclusive. There are different web applications in use. The use of ICT, web applications, software packages, information technologies, devices and platforms, and the internet with all its accompaniments by over 4.95 billion people among the global population is consistently threatened by some expertise users and developers of the sites and applications, as they attack and hack sites and other internet belongings of other users (Okusi, 2023).

The vulnerability of web applications to attacks by cyber criminals, who are known with names like [cyber] attackers, hackers and cyber criminals, causes huge harms and losses to nations,

organizations, groups and individuals alike. Some of the most popular web vulnerabilities are SQL injections, Cross site scripting (XSS) and Cross Site Request Forgery (CSRF). It is reported that XSS presents 40% of attack attempts on web applications, compared to other web related attacks, such as SQLi and CSRF, among others (Rodriguez et al., 2020). Cross-site scripting is when an attacker or hacker injects malicious codes or scripts into a server or on the client side to gain access to critical and important information (Kaur et al., 2023; Okusi, 2023). See the diagrams below:

Fig. 1: How hackers pervert XSS attacks



Source: Okusi (2023) & Kaur et al. (2023)

Cross-site scripting attacks are of national and international concern, because of the threats and losses following the attacks. Essential and confidential data are compromised, stolen or destroyed, putting the nation or organization having the data in serious troubles. The United States of America suffers the devastating effects of cross-site scripting attacks too. For this study, the current attention paid to this critical cyber security concern is very poor. In view of the foregoing, this study engages with class imbalance techniques and cross-site scripting detection and preventive techniques, for which it proposes "Deep Forest" algorithm considered capable of testing for the detection and prevention of cross-site scripting attacks, popularly known in short form as XSS.

Gap in Existing Literatures

Various studies have been carried out, with a view to finding lasting solutions to cyber security threats. The solutions attempts are targeted at detecting, preventing and mitigating cyber-attacks like cross-scripting attacks. Many attempts, including the use of machine learning techniques and other methods, have been introduced for the detection of cross-site scripting attacks, which have produced good results such as obtaining over 90% accuracy scores. Such an accuracy score indicates result-capacity, effectiveness and performance of such algorithms. However, there seems to be a common problem amongst many of the extant studies on the noted subject matter. The problem is that of paying little or no attention to the issue of class imbalance. As Okusi (2023) observes, even when this problem exists in a dataset, there is a good chance of highly obtained biased results. Thus, the need to bridge the gap informed this research. Its novelty rests on the gap.

Aim and Objectives

The aim of this study is to dissect Cross-Site Scripting (XSS) and propose some cyber security techniques for preventing XSS attacks. Its specific objectives are to:

- Investigate cross-site scripting and its variants.
- Describe and show how Deep Forest algorithm and some other cyber security techniques can prevent XSS attacks.
- Examine selected extant studies on XSS for scholarly evidence.

Prevalence of XSS

Cross site scripting (XSS) is a popularly known web based attack, which Marashdih et al. (2019) describe as a code injection attack. It occurs when an attacker injects malicious script code into web applications either on the client side or the server side that enables the attacker to access the victim's vital pieces of information (Okusi, 2023; Chen et al., 2019). Such pieces of information include credit card security details, with which the attackers carry out actions like cookie theft, session hijack, deface the web application, and even cause denial of service (DOS) for the owner(s). Chen et al. (2019) observe that by exploiting the XSS vulnerability in a web application, hackers manipulate enterprise data, including reading, deleting and editing of the data of their victims.

Cross-site scripting attacks have remained very prevalent for several years now. A few scholarly examples here would suffice for the many others. The essence is to justify the claim of this study that XSS is currently very prevalent. Anderson (2020) reports that a cross site scripting attack is linked to a popular hacker group known for stealing confidential credit card information by using credit card skimming methods from unsecured payment portals called Magecartexfiltrated customer records on British Airways website. This hacker group performs this attack by exploiting a cross site scripting vulnerability present on the victim's website, using a Java script library called 'Feedify' which then records customer data (Okusi, 2023; Anderson, 2020; Rodriguez et al., 2020). The attacker tries to avoid being detected by sending the exfiltrated customer data to an external server called 'baways.com'. This attacker group did same to the British Airways (BA). The attack led to a data breach that affected the BA, affecting 380,000 booking transactions between the months of August to September in 2020 (Rodriguez et al., 2020). The BA is the second largest airline carrier of the United Kingdom.

A previous report by Open Web Application Security Project (OWASP) in 2017 has had it that cross-site scripting ranked 7th among the most popular web. As of 2021, XSS ranked 3rd among the top 10 web application security risks (Okusi, 2023). XSS is reported to be leading other web-based cyber-attacks, and 80% of websites are vulnerable to cross-site scripting (XSS) attacks (Reddy, 2022). Also, XSS is responsible for most web application attacks, as it accounts for about 30% of the causes of the other cyber-attacks (Reddy, 2022). The implication of the foregoing reports is that XSS is deadly, pervasive and multifaceted. It causes as well as facilitates high rates of cyber-attacks. This is because apart from causing direct attacks, it also contributes to other attacks. Thus, XSS is posing serious threats to organizations and nations.

Wedging a strong systemic war against it is imperative. Research, such as this current work, is a viable way of contributing to wedging war against the scourge. The US, the world's power, ought to make concerted efforts to prevent XSS attacks on its airlines and other organizations, learning from the hard lessons of the XSS attacks on BA of the UK. The WordPress Security Learning once

averred that if a single code can be used to handle SQL injection and XSS vulnerabilities, the code would solve 65% of web apps and websites vulnerabilities (Abikoye et al., 2020). The opinion suggests that there is the dire need for tactical technology-based measure(s) to be deployed against the scourge of XSS attacks. Chaudhary's et al. (2020) study shows that many researchers have done a lot of work in finding a tangible solution to XSS attacks, proposing several defensive techniques and solutions, but most of them suffer from high rates of false positives and false negatives.

The Chaudhary's et al. (2020) study elaborates on XSS attacks on social media, showing some statistics of XSS attacks on social networking sites (SNS). There have been XSS attacks on:

- UK Parliament website attack in 2014, with disinformation being the major effect of the attack.
- Yahoo web site, hacking its central account in 2013.
- Hotmail website in 2011, hijacking the website of Hotmail and causing a lot of harms and losses to the company.

Considering the severity of XSS attacks, it is imperative to take up a study of this kind in order look at the scourge, draw deserving attention to it, and weigh the effects of the attacks as the catalysts of the prompt actions and measures needed against the scourge. A study of this kind is imperative because it is of national and international concerns and relevance. As it is, the relevance of this work to the US cannot be over-emphasized, because it makes a call to US stakeholders in various government parastatals and organizations of both private and public sectors to rise to the challenge of tackling the dreaded menace of XSS.

The study contributes to ways of detecting and preventing XSS attacks, which are of great benefits to the US and other parts of the globe. It emphasizes the importance of taking proactive measures against XSS to avert its attacks and devastating effects. Doing so (the needful) proactively would undoubtedly help guarantee cyber security, and prevent privacy invasion, threats to user data, credentials breach, impersonation, DOS, etc. By making valuable contributions to the discourses on cyber security in general and XSS and class imbalance in particular, the study is an attempt to reduce the prevalence of XSS attacks and solve some salient problems in the contemporary globalized and digitalized society.

Types of XSS

Cross site scripting attacks (XSS) is said to be classified by many as: Stored, Reflected, and DOM (OWASP, 2017). For Marashdihet al. (2019), the classification of XSS into three is overlapping in reality. Thus, XSS is rather of two main classes viz: Server-side XSS and Client-side XSS vulnerabilities. To them, the client-side XSS is the DOM-based XSS and the server side XSS consists of the Reflected or Non-persistent XSS and the Stored or Persistent XSS (Marashdihet al., 2019). A brief on each of the aforementioned three (3) would suffice hereafter.

Reflected Cross-Site Scripting (XSS)

In a reflected XSS attack, the victim is lured and tricked into clicking a malicious url, which contains malicious script. When a victim clicks on this link, the result gotten from the website

contains the malicious code that is sent from the server. This attack uses the information retrieved from the user to generate a response. This attack can be carried out through links, forms, cookies, etc. Once the attack is successful, all the victims' traffic gets redirected to the hacker. The Figure 1 below shows the reflected cross attack.

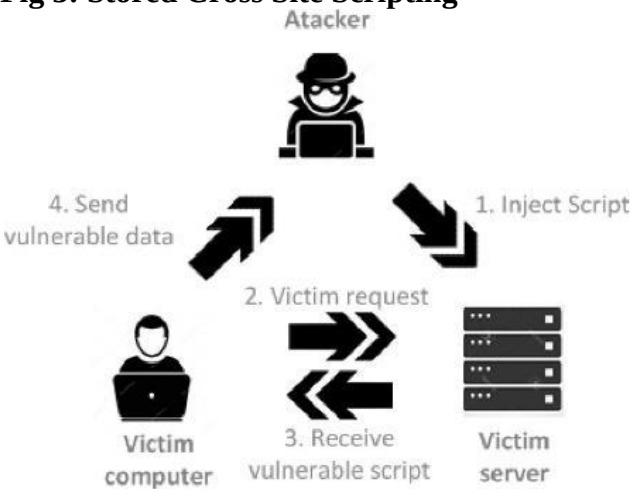
Figure 2: Reflected Cross Attack

Source: Okusi, (2023, p. 15)

Stored Cross Site Scripting (XSS)

In a stored XSS attack, the web application is exploited by injecting malicious code into it and then stored on a web server. The malicious payload is already on the server and will be executed when a victim visits this web application. This attack can occur when user input is stored on the target server like in a database, message forum, comment box, etc., and a victim then unknowingly retrieves the stored data from the target server which initiates the attack.

Fig 3: Stored Cross Site Scripting

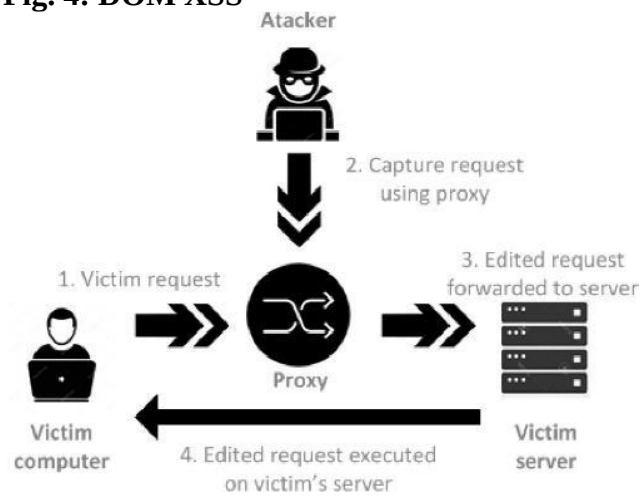


Source: Okusi, (2023, p. 15)

DOM-Based Cross Site Scripting (XSS)

In a DOM based XSS attack, the attack happens on the client side, such as the browser. This means that during the attack, the entire dataflow takes places in the browser. A Document Object Model (DOM) is a programming interface for web documents. It is a tree-like structure that represents a page where programs can change structure, style and content. DOM XSS is an attack where the victim is tricked to click on a malicious URL that enables the malicious script to be executed at a certain point when a web page is loaded on to the victim's computer. This attempt is similar to the reflected XSS, since it involves the victim clicking on the malicious link. The only difference is that this attack does not require the involvement of a server. Consider the daigrammed representation of DOM below:

Fig. 4: DOM XSS



Source: Okusi, (2023, p. 15)

Related Literatures

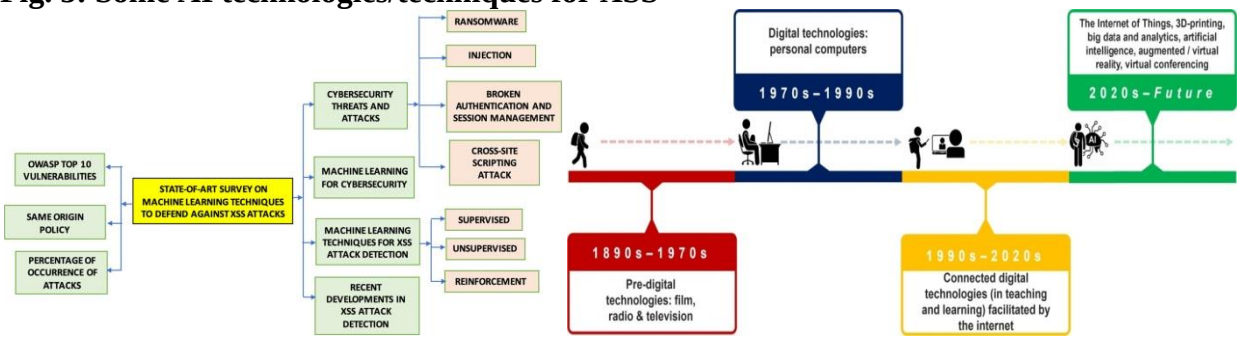
Kaur et al. (2023) explore machine learning and neural network-based XSS attack detection techniques, such as deep neural networks, decision trees, web-log-based detection models, and prove them to be viable mechanisms for detecting and preventing XSS attacks. The study charges future researches to work more on techniques for improving detection and prevention of XSS attacks. Thus, the present study is an attempt in that direction to meet that scholarly obligation. The study lends credence to the present one by showing that XSS can be detected using some AI techniques. The detection would undoubtedly lead to the prevention. By detecting and preventing it, the threats of XSS to cyber security would reduce significantly, if not eradicated completely.

Similarly, Okusi (2023) analyzes cross-site scripting, identifying some of its preventive techniques. The study indicates that web application is being highly attacked, because of the vulnerabilities that exist on the online space while using the modern technologies. Thus, in order to have a secure or safe web application, preventive techniques have to be devised and utilized accordingly. The study is thus an attempt in that direction to contribute to quelling web application attacks, which are some threats to cyber security. The work by Okusi (2023) highlights the use of cross-site scripting, as it notes that cross-site scripting attack may be used for storing unauthorized cookies or other data in the browser session of the potential victim, with the intent of injecting script into web pages. The research proposes the leveraging of "DEEP FOREST", a developed but

yet to use machine learning algorithm, and four class imbalance techniques to address the risks of cross-site scripting attacks. The study relates to the present one, as it concerns itself with cross-site scripting and preventive cyber security techniques with which the cyber security threats arising from XSS can be prevented as well as reduced.

The study carried out by Zhang et al. (2019) proposes a new approach to detecting XSS attacks, using information from multiple stages. The approach calls for the use of Word2vec, a technique of Natural Language Processing (NLP). NLP is an integral part or a phase of Artificial Intelligence (AI). By proposing the use of this technique to detect XSS attacks, the study undoubtedly proposes as well as affirms the leveraging of AI for XSS detection and control or prevention. It means that AI can be used to proffer solution to XSS attacks, at least by reducing the menace of XSS. As Kaur et al. (2023) show in their survey flowchart of AI in combating XSS, the following diagrams offer symbolic insights to AI technologies that can be leveraged for XSS attacks and other cyber security threats:

Fig. 5: Some AI technologies/techniques for XSS



Source: Kaur et al. (2023)

Again, Zhang's et al. (2019) study demonstrates that Word2vec is a technique used to learn word embedding from a text corpus. The study also suggests the use of word2vec to streamline NLP so as to use the Continuous Bag of Words (CBOW) instead of the Skip-gram technique, because the CBOW is more time-efficient. It also introduces the Gaussian Mixture Model (GMM) as a mechanism for detecting XSS. It is a dual model approach. While one model is proposed for characterizing XSS attacks, the other is proposed for characterizing normal web transactions. Then, the predictions obtained from these two models are thereby integrated to produce a more accurate result. In the research, the XSS dataset obtained was 45,884 and the normal transaction dataset obtained was 34,561. During the model fitting stage, 35,884 and 24,561 were used for training the model for both the XSS and the normal web transactions respectively. Alternatively, 10,000 datasets were utilized for testing both models equally (Zhang et al., 2019). The results obtained during the research were good, but the large difference in the dataset utilization for training indicates a class imbalance in the approach. Regrettably, the researchers disregard the imbalance, which matters to the present study.

Deep Forest (DF) is more or less like Deep Neural Networks (DNN). He also stated that DF is similar to random forest (RF), because RF contains many decision-trees. Deep forest is made up of multiple layers in its architectural design and it is a collection of multiple random forests in an

organized manner. The DF model is more user friendly and easier to train (Okusi, 2023). Unlike DNN, where a large amount of training data is required, the DF algorithm can function excellently with small scale data. This point highlights why DF can be used to prevent XSS. DF algorithm also does not have many hyper-parameters, unlike DNN. However, the DF model itself performs excellently without the use of hyper-parameters. Yet, if hyper-parameter tuning techniques are employed, the DF model becomes even more robust and efficient in its performance.

Thakkar and Lohiya (2021) prove that Machine Learning techniques can be leveraged for cyber security, including XSS attacks. According to Zhang et al. (2021), the interest in AI rests on its advanced computing power, for which it solves many problems, such as language analysis, speech recognition, machine translation, robotics, and cyber defense, among others. The study by Jian-hua Li (2021) makes a detailed systematic review of the many different literatures on the role of Machine Learning (ML) and Deep Learning (DL) algorithms in the internet world. It indicates that AI technologies also face cyber-attacks. The study goes on to propose the encrypting of the deep neural network as a way of preventing cyber-attacks on AI technologies. According to the study, the major limitation of Support Vector Machine (SVM) as well as other like AI technologies is that intensive time-consuming training is required. Nevertheless, with AVL Tree, training and attack detection time can be reduced and improved.

Furthermore, studies argue against the possibility of using Deep Learning, Machine Learning and dynamic analysis-based approaches to quelling the cyber security threats posed by XSS attacks (Tariq et al. 2021; Kaur & Singh, 2019; Zhou & Wang, 2019). These cited authors argue that ML techniques are inefficient in detecting and preventing XSS attacks. On the other hand, Zhang et al. (2022), Dixit and Silakari (2021), Luo et al. (2021), Onan and Tocoglu (2021), Pavan Kumar et al. (2021), and Onan (2019b), among others, demonstrate that Deep Learning techniques can be leveraged for detecting and preventing XSS attacks in particular and other cyber security threats in general.

It is imperative note that DL techniques are categorized into: supervised, unsupervised, and reinforced (Pavan Kumar et al. 2021). The implication of the foregoing two sets of views on the use of AI technologies and techniques to quell XSS as well as other cyber security is that despite being effective or result-oriented, AI-based techniques also have pitfalls which make them inefficient in some regards. Nevertheless, more studies than not affirm the potentials of AI technologies and techniques in tackling cyber security challenges. That is, regardless of their shortcomings, they are largely capable of tackling cyber security challenges when deployed rightly and the attendant constraints to their efficacy are ameliorated.

Methodology

The study combines exploratory, comparative and experimental analyses. The mode of data collection for this research is secondary. The dataset is collected from github from an upload created by fmereani in a work similar to our research. The dataset retrieved is imbalance because the data tends towards the benign class. The dataset collected contains three different types of data combined: the payload, the train dataset and the test dataset. The payload, XSSS training dataset and XSS testing dataset contain 43218, 19123 and 24097 data respectively. The adopted approach

utilized a dataset obtained from github alongside an already developed algorithm called Deep Forest to test its ability to detect and as well prevent XSS attacks. The approach addresses the issue of class imbalance using some data resampling techniques and libraries to solve the problem. The results obtained are evaluated and conclusion is drawn with graphical representations like confusion matrix and graphs, and according to the following metrics: true positive, false positive, precision, accuracy score, etc.

True positive (TP): TP is the proportion of correctly and positively classified samples. This is when the machine learning model correctly identifies malicious script or url in the dataset.

$$\frac{\mathrm{TPR} = \mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})}$$

False positive (FP): FP is the percentage of samples that are misidentified as positive. This is when the machine learning model incorrectly identifies malicious scripts or samples as benign

$$\frac{\mathrm{FPR} = \mathrm{FP}}{(\mathrm{FP} + \mathrm{TN})}$$

True Negative (TN): True negatives are those samples that were appropriately categorized as being negative. This is when the machine learning model correctly identifies malicious urls or scripts as malicious.

$$\frac{\mathrm{TN} = \mathrm{TN}}{(\mathrm{TN} + \mathrm{FP})}$$

False negative (FN): False negative relates to how many samples had their status misclassified as negative. This is when the model incorrectly identifies benign url or scripts as malicious.

$$\frac{\mathrm{FN} = \mathrm{FN}}{(\mathrm{FN} + \mathrm{TP})}$$

Precision (P): The proportion of accurately anticipated positive observations to all of the predicted observations is known as precision. This is the rate at which the correctly detected malicious scripts are actually malicious.

$$\frac{\mathrm{P} = \mathrm{TP}}{(\mathrm{TP} + \mathrm{FP})}$$

Recall: The recall measures how many correctly predicted and favorable observations were compared to all of the actual class observations.

$$\frac{\mathrm{R} = \mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})}$$

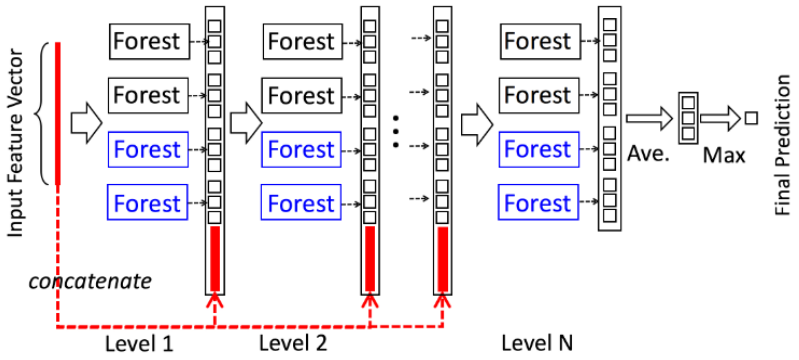
F1 score (F1): Precision and Recall are weighted averaged to determine the F1 score and because of this, both false positives and false negatives are considered.

$$\frac{\mathrm{F1} = 2 * \mathrm{R} * \mathrm{P}}{(\mathrm{R} + \mathrm{P})}$$

Accuracy score: Accuracy score is the score obtained after the machine learning model has successfully carried out a prediction. It indicates a fraction of the dataset that it correctly predicted. It is usually in percentage (%).

Furthermore, the study proposes Cascade Forest classifier to test for the detection of XSS attacks and to measure its performance when compared to other algorithms, after eliminating the issue of class imbalance. The structure of DF algorithm is shown graphically below:

Fig. 6: DF algorithm structure



Source: Okusi (2023)

The test approach for the detection of XSS attacks using the DF algorithm involves several steps. The first step is to check both train and test dataset. Next, data preprocessing is done to the already processed data using the LabelEncoder. The class imbalance techniques are applied, and then proceeded to identifying and selecting the most important features using ExtraTreeRegressor model. The most important features from the previous step are fed to train the DF classifier. For a better result, some hyper-parameter tunings to the model are applied. The steps in each stage for this research are tabulated hereunder:

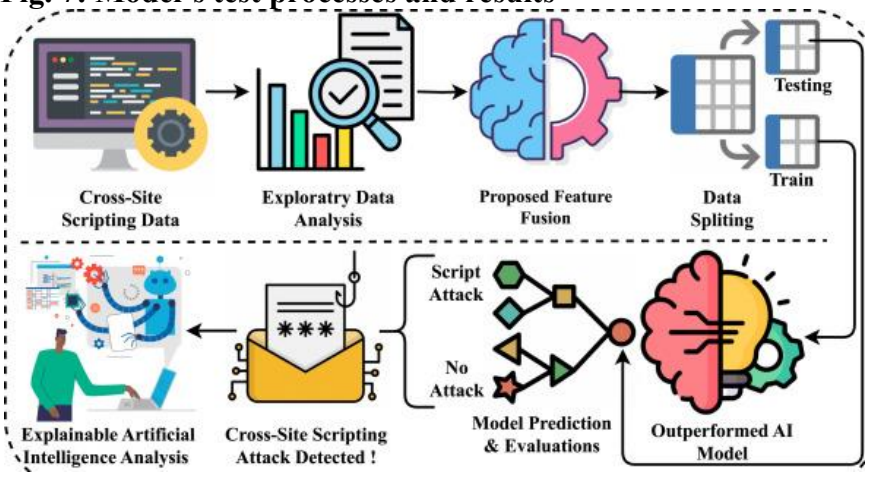
Table 1: Steps and actions taken

Steps	**Actions**
Step 1	Check for class imbalance and apply further data preprocessing using the label encoder
Step 2	Apply class imbalance techniques
Step 3	Feature selection using ExtraTreeRegressor model
Step 4	Train and test Deep Forest model
Step 5	Apply hyper-parameter tuning
Step 6	Final result

Source: Author's Computation, 2024

From the above table, it is realized that to attain or arrive at the results, the following steps have to be taken or followed: load dataset, check dataset for class imbalance, further data preprocessing, selection of important features, train and test DF model, and apply hyper-parameter tuning. Since after the application of hyper-parameters, results are realized, it means that the steps begin with loading dataset and end with apply hyper-parameter tuning. Consider diagram below:

Fig. 7: Model's test processes and results



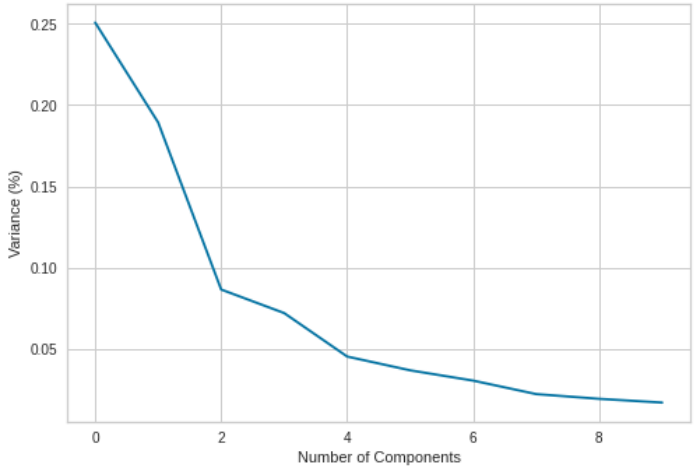
Source: Adapted from Okusi (2023)

Data Presentation and Analysis

Principal component analysis (PCA)

Principal component analysis (PCA) is a Dimensionality Reduction algorithm. The PCA algorithm performs a linear mapping of the data to a lower dimensional space in such a way that the variance of the data in the low dimensional space is maximized. This research proposes the use of PCA because it helps to reduce information loss and also increases interpretability.

Fig. 8: PCA performance graph



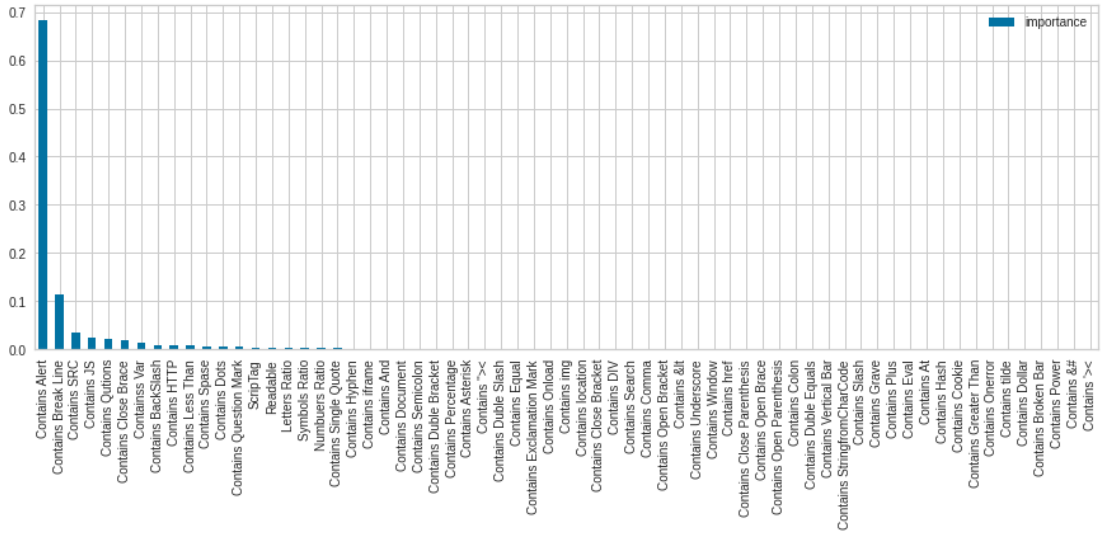
Source: Author's computation, 2024

Feature selection

Feature selection is an input variable reduction technique that helps to eliminate noise and overfitting of the model by selecting and utilizing only the relevant data in the dataset. The train and test datasets collected are in their processed form, containing 66 columns of features extracted from the payload dataset. For this stage, a model called ExtraTreeRegressor is adopted. The ExtreTreeregressor is used to fit it and transform the training data. The model is then used to predict the feature importance of each attribute in the training data. The feature importance is then plotted as follows:

Table 2: Attributes and data training

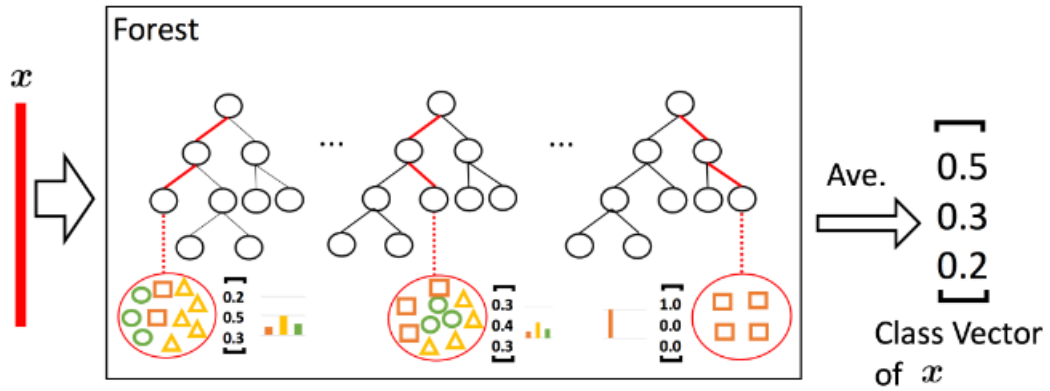
S/N	Feature Name	Importance
1	Contains Alert	0681968
2	Contains Break Line	0.113957
3	Contains SRC	0.033974
4	Contains JS	0.024444
5	Contains Qutions	0.022838
6	Contains Close Brace	0.018409
7	Contains Var	0.014161
8	Contains BackSlash	0.009760
9	Contains HTTP	0.008153
10	Contains Less Than	0.008034
11	Contains Space	0.007255
12	Contains Dots	0.006862
13	Contains Question Mark	0.006575
14	ScripTag	0.003110
15	Symbols Ratio	0.003020
16	Readable	0.002970
17	Letters Ratio	0.002898
18	Numbers Ratio	0.002454
19	Contains Single Quote	0.002181
20	Contains Hyphen	0.001812



Cascade Forest Classifier

This research proposes the use of Cascade Forest (CF) because cascade forest is a part of the deep forest library, an ensemble learning algorithm, used for classification tasks. CF contains different levels and each levels in the CF receives input information of processed features from the preceding level and sends its processing result to the next level. The cascade forest is an ensemble of decision trees. An ensemble is a ML paradigm that is used to describe a combination of trained multiple classifiers that are then used to make predictions with greater accuracy and stability than a single decision tree.

Fig. 9: CF structure



Source: Author's computation, 2024

Hyper-Parameter Tuning

Hyper-parameter tuning is a technique used to maximize a ML algorithm for a better performance using hyper-parameters. Hyper-parameters are model-specific. So, the parameters used for one model will produce a different result when used for another model. Hyper-parameters are explicitly defined and used by the model in its learning process. This study's approach manually defines

some of these hyper-parameters and then fits the model with the parameters. The model then predicts the labels for the test data that the DF model of this study utilizes in its leaning process to maximize its performance and obtain a result with minimal loss of data. The accuracy of the model is then printed along with the classification report. Following were the best parameter set-up for the model:

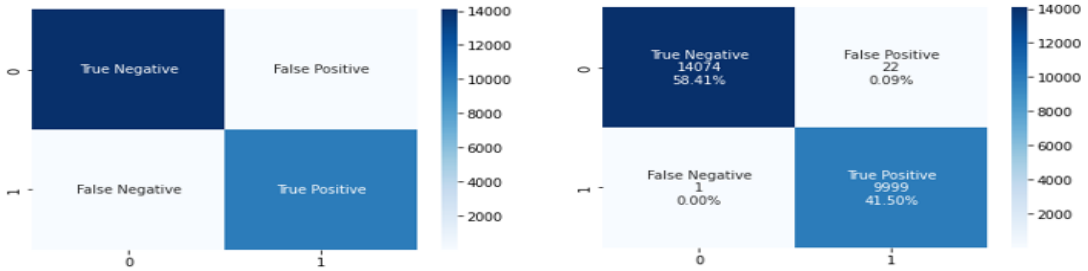
{'max_depth': 10,

'min_samples_leaf': 100,

'min_samples_split': 200,

'n_estimators': 20}

Fig. 10: Model metrics



Source: Author's computation, 2024

Fig. 11: Result of normal parameter of the model

Testing Accuracy: 99.905 %

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14075
1	1.00	1.00	1.00	10021
accuracy			1.00	24096
macro avg	1.00	1.00	1.00	24096
weighted avg	1.00	1.00	1.00	24096

Fig. 12: Result of the hyper-parameter tuning of the model

Testing Accuracy: 99.896 %

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14113
1	1.00	1.00	1.00	9983
accuracy			1.00	24096
macro avg	1.00	1.00	1.00	24096
weighted avg	1.00	1.00	1.00	24096

Fig. 13: Result of the hyper-parameter tuning of the model with random oversampling

```
Testing Accuracy: 99.822 %
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     14063
           1       1.00      1.00      1.00     10033

    accuracy                           1.00     24096
   macro avg       1.00      1.00      1.00     24096
weighted avg       1.00      1.00      1.00     24096
```

Fig. 14: Result of the hyper-parameter tuning the model with random undersampling

```
Testing Accuracy: 99.797 %
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     14135
           1       1.00      1.00      1.00      9961

    accuracy                           1.00     24096
   macro avg       1.00      1.00      1.00     24096
weighted avg       1.00      1.00      1.00     24096
```

Fig. 15: Result of the hyper-parameter tuning the model with Smote

```
Testing Accuracy: 99.863 %
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     14077
           1       1.00      1.00      1.00     10019

    accuracy                           1.00     24096
   macro avg       1.00      1.00      1.00     24096
weighted avg       1.00      1.00      1.00     24096
```

Fig. 16: Result of the hyper-parameter tuning of the model with Near-miss

```
Testing Accuracy: 90.496 %
              precision    recall  f1-score   support

           0       0.84      1.00      0.91     11876
           1       1.00      0.82      0.90     12220

    accuracy                           0.90     24096
   macro avg       0.92      0.91      0.90     24096
weighted avg       0.92      0.90      0.90     24096
```

Source: Author's computation, 2024

Discussion of Results

The result of the confusion matrix for the first test shows that the Deep Forest (DP) model is capable of correctly identifying 14074 samples of the true negative, 9999 samples of the true positives, 1 sample of false negative and 22 samples of false positive. Very little concern is shown for the misclassification of the false negative, but only 22 malicious samples were wrongly

classified as benign. The second test the same imbalanced dataset is used but some hyper-parameter tuning is performed on the study's model, for which an accuracy score of 99.896% is achieved. In the third test, a balanced dataset is utilized using the random over sampling technique together with hyper parameter and an accuracy score of 99.8% is achieved. In the fourth test, a balanced dataset is utilized using the random under sampling technique together with hyper parameter and an accuracy score of 99.7% is achieved. In the fifth test, a balanced dataset is utilized using the SMOTE sampling technique together with hyper parameter and an accuracy score of 99.8% is achieved. In the sixth test, a balanced dataset is utilized using the random Near miss sampling technique together with hyper parameter and an accuracy score of 90.4% is achieved.

The results prove the proposed model to be better than in performance with our imbalanced dataset. On the other hand, once the class imbalance techniques are applied, its performance dropped. Nevertheless, the difference in performance is by a minimal margin. The most significant change is in the test, with the near-miss sampling technique that is a type of down sampling technique. From the results above, it is quite understandable that the tests having over sampling techniques applied had a better result than the other two tests where down sampling were applied. Thus, it is logical to posit that the proposed model performed well in all cases of the tests carried out. This assertion is made in view of the results of some previous studies. In a study done by Akaishi and Uda (2019) support vector machine(SVM) achieved a result where accuracy is 0.9928, precision 99.80, recall 0.9875 f1 score 0.9937.

The results of Banerjee's et al. (2020) study reveal that the random forest achieved accuracy 0.98, precision 0.983, recall 0.99, f-measure 0.955. Also, Kascheev and Olenchikova's (2020) study show that 98.91 accuracy, 0.9919 precision, 0.9370 recall and 0.9590 f1-score. Finally, the result of the study's approach after applying class imbalance techniques to solve the issue of class imbalance in its dataset, compared the other models, indicates that the model performed slightly better than them, more evidentially in the precision, recall and f1 score. The accuracy in scores varied, but the proposed model is able to edge the SVM, the random forest and decision trees performed just the same as the DF model, when compared to our tests with the over sampling techniques.

Conclusion

So far, this research has explored some XSS preventive techniques. It also presents an algorithm that has been developed but has not been tested for the detection of cross-site scripting attack. Its approach is to eliminate the issue of class imbalance to test the performance of the model. It has shown that class imbalance can be eliminated using DF model and several techniques, including the use of data resampling techniques. Using some class imbalance technique libraries, the issue of class imbalance was eliminated and the result of that is used to train and test the model proposed by this study. The DF model has been proven to be capable of detecting and preventing XSS attacks.

Therefore, just as Deep Learning and other AI technologies are proven by previous studies to be viable means of detecting and preventing XSS and other cyber-attacks, DF model is likewise and even better in some regards. Regarding performance, after solving the issue of class imbalance,

the DF algorithm has shown that it can correctly detect XSS attacks when compared with other algorithms like random forest, support vector machine and decision tree, though the difference in performance is not so much. Indeed, the proposed model performed better in other metrics asides the accuracy score, including performing better than the SVM.

Recommendations

From the results achieved, there are several other areas that can be improved on. In future, the developed approach will be to explore using several other class imbalance techniques and larger dataset. Also, improvement can be done in the area of hyper-parameter tuning techniques, such as grid search, random search and others. The DF algorithm can further be enhanced for a better XSS detection and prevention by applying the multi grained cascade forest (GcForest). Although DF model produced great results, when tested with a balanced dataset using the oversampling techniques, it is believed that the two sampling techniques overfit the model.

References

Abikoye, O. C., Abubakar, A., Dokoro, A. H., Akande, O. N. & Kayode, A. A. (2020). A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string algorithm. *EURASIP Journal on Information Security, vol.2020*, 1-14.

Akaishi, S. & Uda, R. (2019). Classification of XSS attacks by machine learning with frequency of appearance and co-occurrence. *53rd Annual Conference* on information sciences and systems (CISS), *IEEE*, 1-6.

Anderson, B. (2020, December). "3 dangerous cross-site scripting attacks of the last decade." *ReadWrite.1*. Available from: https://readwrite.com/3-dangerous-cross-site-scripting-attacks-of-the-last-decade.

Banerjee, R., Baksi, A., Singh, N. & Bishnu, S. K. (2020). Detection of XSS in web applications using machine learning classifiers. *4th International Conference* on electronics, materials engineering & nano-Technology (IEMENTech), *IEEE*, 1–5.

Chaudhary, P., Gupta, B. B. & Gupta, S. (2016). Cross-site scripting (XSS) worms in online social network (OSN): Taxonomy and defensive mechanisms. *3rd International Conference on Computing for Sustainable Global Development (INDIACom*), 2131–2136.

Chen, X., Li, M., Jiang, Y. U. & Sun, Y. (2019). A comparison of machine learning algorithms for detecting Xss attacks. *Artificial Intelligence and Security*, 11635, 214-224.

Dixit, P. & Silakari, S. (2021). Deep learning algorithms for cyber security applications: A technological and status review. *Computer Sci. Rev.*, 39, 100317. https://doi.org/10.1016/J. COSREV.2020.100317

Jian-Hua, L. (2021). Cyber security meets machine learning. In cyber security meets machine learning. *Springer Singapore*. https:// doi. org/ 10. 1007/ 978- 981- 33- 6726-5

Kascheev, S. & Olenchikova, T. (2020). The detecting cross-site scripting (XSS) using machine learning methods. Global Smart Industry Conference (GloSIC), *IEEE*, 265–270.

Kaur, J. Garg, U. & Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review. *Artificial Intelligence Review*, https://doi.org/10.1007/s10462-023-10433-3

Kaur, S. & Singh, M. (2019). Hybrid intrusion detection and signature generation using deep recurrent neural networks. *Neural Computer App.*, 32(12), 7859–7877. https:// doi. org/ 10. 1007/ S00521- 019- 04187-9

Luo, C., Tan, Z., Min, G., Gan, J., Shi, W. & Tian, Z. (2021). A novel web attack detection system for internet of things via ensemble classification. *IEEE Trans. Industry Inf.*, 17(8), 5810–5818. https:// doi. org/ 10. 1109/ TII. 2020.3038761

Marashdih, A. W., Zaaba, Z. F., Suwais, K. & Mohd, N. A. (2019). Web application security: An investigation on static analysis with other algorithms to detect cross site scripting. *Procedia computer science*, 161, 1173–1181.

Okusi, O. A. (2023). An analysis of cross-site scripting and its preventive techniques. MSc. cyber security: CSCT Masters Project. Department of Computer Science and Creative Technologies.

Onan, A. & Tocoglu, M. A. (2021). A term weighted neural language model and stacked bidirectional LSTM based framework for sarcasm identification. *IEEE Access*, 9, 7701–7722. https://doi.org/ 10.1109/ACCESS. 2021. 30497 34

Onan, A. (2019b). Topic-enriched word embeddings for sarcasm identification. *Adv. Intell. Syst. Comput*., 984, 293–304. https:// doi.org/10.1007/978-3-030-19807-7_29

OWASP (2017). OWASP top ten. *OWASP. https://owasp.org/*

Pavan, K. P., Jaya, T. & Rajendran, V. (2021). SI-BBA— a novel phishing website detection based on swarm intelligence with deep learning. *Mater Today*. https:// doi. org/ 10. 1016/J. MATPR. 2021. 07. 178

Reddy, H. B. S. (2022). A proposal for emerging gaps in finding firm solutions for cross site scripting attacks on web applications. *International Journal of Research Publications and Reviews, vol.3, iss.7*. 3928-3985. DOI 10.55248/gengpi.2022.3.7.43

Rodríguez, G. E., Torres, J. G., Flores, P. & Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. *Computer networks*. 166 (106960), 106960

Tariq, I., Sindhu, M. A., Abbasi, R. A., Khattak, A. S., Maqbool, O. & Siddiqui, G. F. (2021). Resolving crosssite scripting attacks thriough genetic algorithm and reinforcement learning. *Exp Syst Appl, 168*. https:// doi. org/ 10.1016/J. ESWA.2020.114386

Thakkar, A. & Lohiya, R. (2021). A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges and future research

directions. *Artificial Intell Rev, 55(1*), 453–563. https:// doi. org/ 10. 1007/ S10462-021-10037-9

Zhang, G., Liu, B., Zhu, T., Zhou, A. & Zhou, W. (2022). Visual privacy attacks and defenses in deep learning: A survey. *Artif Intell Rev, 1–55*. https:// doi. org/ 10. 1007/ S10462-021- 10123-Y

Zhang, J., Jou, Y. & Li, X. (2019). Cross-site scripting (XSS) detection integrating evidences in multiple stages. *Proceedings of the 52nd Hawaii International Conference on System Sciences, 7174*.

Zhang, Z., Ning, H., Shi, F., Farha, F., Xu, Y., Xu, J., Zhang, F., & Choo, K. K. R. (2021). Artificial intelligence in cyber security: Research advances, challenges, and opportunities. *Artif Intell Rev, 55(2*), 1029–1053. https:// doi.org/10.1007/S10462-021- 09976-0

Zhou, Y. & Wang, P. (2019). An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence. *Computer Security, 82*, 261–269. https:// doi. org/ 10.1016/J.COSE.2018.12.016