National College of Ireland

Classification of Malicious Web Code Using Deep Learning

MSc Research Project
Data Analytics

Manoj Kumar Selvam
x17115663

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Manoj Kumar Selvam
Student ID:	x17115663
Programme:	Data Analytics
Year:	2018
Module:	MSc Research Project
Lecturer:	Vikas Tomer
Submission Due Date:	13/08/2018
Project Title:	Classification of Malicious Web Code Using Deep Learning
Word Count:	5,793

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	13th August 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Classification of Malicious Web Code Using Deep Learning

Manoj Kumar Selvam
x17115663
MSc Research Project in Data Analytics

13th August 2018

Abstract

As the world moves towards web, there have been a lot of attacks which was carried out in web applications. Among those, we have the XSS(Cross-site scripting) attack which is considered as one of the top attack as per OWASP (Open Web-Application Security Project). The attacker uses a sophisticated method of injecting malicious code into the web application through web forms or request parameters, which is the stored in the server and later executed when the user visits the vulnerable page. In this paper we are using deep learning approach to identify the stored XSS vulnerabilities and to detect them, in order to prevent such malicious attacks. The paper discusses in detail the methodology used to apply deep machine learning methods and helps us identify malicious and non malicious web code. The experiment conducted using 11,000 labelled samples, provided the accuracy of 98% for the proposed architecture.

1 Introduction

In today's interconnected world, everything and everyone around the globe is connected through the internet. This gives access to a world of opportunities and businesses to thrive and grow, providing a global presence. Hence, there is a requirement of emerging web application development across all domains. These days there is an increase in the number of web-applications across the market. Any such demand also has to be dealt with improved security, and for such surge in web applications, the developers have an increased responsibility to secure these developed web applications. The web applications are at the risk of hidden vulnerabilities. According to OWASP (Open Web-Application Security Project), cross-site scripting attacks, rank highest among all other attacks. Cross-Site Scripting (XSS) is a sophisticated attack done by hackers, where a malicious script is injected into the web applications through web forms or web request and later, saved on the server. As soon as end-user access the infected webpage on the web-application, the attacks get actuated and user-data is stolen. The attacks are commenced by the attacker using Javascript and HTML. What makes XSS attacks hard is due to their peculiar hidden presence. The malicious Javascript remains undetected for longer to an unknown time.

Web applications in general are prone to different kind of attacks. However, trying to prevent these attacks may just not stop hackers from utilising any existing or new loopholes to inject these attacks on the web applications. When security is developed to tackle these attacks, it becomes imperative to keep revising the current methodologies and to keep the web applications secure. These web applications, if hacked may prove a costly affair. One such example, though not a web application security attack, but a loophole where data of millions was leaked from Facebook and available at a 3rd party company without the user's consent. This keeps the end user on a constant lookout for the security of the webpage, to know know how secure and trusted a webpage is by other users, and hense are hesitant to provide or give out any information online on these web applications.

XSS attacks executes javascript code on client's browser through which the attackers steal user information. There are different types of XSS attacks. One such type is Reflected XSS, where the malicious code will be passed in the request URL. This gets executed when user access those infected URLs. DOM-based XSS is another form where the vulnerability originates from client side instead of server side.In Persistent XSS attack which is also called as Stored XSS attack, the malicious code is injected in website's database or file system. When user accesses the page which is affected, the code gets executed. This paper deals with Stored/Persistent XSS attack and methods to classify vulnerable and non-vulnerable code using machine learning approach.

Lot of existing work has been done to identify the XSS vulnerability from web code. Static analysis of code is one of them, where the source code is reviewed before the execution. This technique can guarantee the prevention of existing vulnerability but the identification process takes much time. Sometimes its slow, which fails to provide the accurate result. Dynamic analysis is another method where the system understands the behaviour of existing malware. In this method the rules were defined by understanding the existing attacks and a system is developed to prevent the vulnerability using those rules. However, this method fails to identify any new attacks because of the system design. Whenever a new vulnerability is found, the system is modified to block certain attacks by defining new rules.

Machine learning method of identifying the vulnerability is another method, Where the machine learning classifier is trained to identify vulnerable and non-vulnerable code. The features are extracted from vulnerable and non-vulnerable code by using different NLP techniques like TFID, Bag of words, etc. The extracted features are used to train the model for classification using various algorithms like KNN and Random Forest. This method has improved the capability of identifying vulnerable code with higher accuracy and precision rate. However, in this technique the feature extraction method used is not reliable as we are handling the program code written in HTML and Javascript. A program code will have lot of repeated keywords using techniques like word frequency, Bag of words, etc., resulting in a feature vector of all the keywords and its frequency, Which is not a viable method for extracting the features.

The proposed approach deals with a new way of extracting meaningful features from the web code for training the classifier. The collected features were tested with Convolution Neural Network (CNN) classifiers and its performance is compared with K-Nearest Neighbors (KNN), Naive Bayes and Random Forest (RF). The Cross Industry Standard Process for Data Mining (CRISP-DM) methodology is used for this study. The rest of the research report is as follows: Literature Review forms the second part of the project. The methodology is then discussed along with the architectural design. Later the models are

developed, designed and implemented presenting the results of the research experiment. Finally, the research is concluded with the aforementioned model experiment findings and any future work.

2 Related Work

Security anomaly detection through machine learning is not new. There has been extensive research done on how we can detect such anomalies in the form of java scripts, html scripts etc., within the web code. However, the approaches and methodologies used were particularly with the problem the researchers were trying to solve. One such research is where a static malicious javascript is detected through SVM (Support Vector Machine) by Likarish et al. (2009). In this research, it was proposed to use the non-linear, static, SVM (Support Vector Machine) based analysis method. The accuracy attained was quite high at 94.38% for the experiment conducted on the static script with a low false alarm, as compared to other models. However, the author Rathore et al. (2017) falls short in providing reasoning for using a non linear SVM (Support Vector Machine) approach, mentioning it as a complex data set. Along with this, we are unaware on why the dynamic script analysis was not a part of the research.

The cross site scripting (XSS) attack detection using machine learning by Fawaz and Jacob (2018) and Gupta et al. (2015) explains how a combination of language syntax and behavioral features provide a greater accuracy on testing datasets. The author explains how the XSS and SQL injection attacks are one of the top most and popular choice for stealing information. It also showcases how obfuscation occurs and the code can be changed or modified in the form of both benign and malicious intents. The study boasts of being the first to use Random Forests as a classifier to detect the cross site scripting (XSS) attacks. Malviya et al. (2013) proposed a method where structural features, behavioral features and classifiers were implemented by converting them into binary measures instead of the traditional approach of the weighted measures being taken into consideration. This provided a very high rate of accuracy with almost all models. But at the same time, we do not have a reasoning for this particular approach being taken to obtain this high accuracy of more than 95% across models such as k-NN, Random Forests, Linear and Polynomial SVM. Except the Linear SVM model, all other models reach an accuracy of or more than 99.50% on the testing data sets.

Ghaffarian and Shahriari (2017) discuss about various methods of discovering XSS vulnarablity. Where Word frequency methoed (TFID), identification using strurcural and functional behaviour and Abstract syntax tree methods are popuplar methods used for classify vulnerable and non-vulnerable codes. However these methods are not good in understanding the pattern. which makes the classifier ineffective to classify new attacks.

A similar research was carried out earlier by Vishnu and Jevitha (2014), where the author used the URL and JavaScript-based features to train the algorithms specific to these.The models built were the Naive Bayes, Support Vector Machines and J48 Random Forest. However, the results for these experiments were evaluated based on the FPR (False Positive Rate), TPR (True Positive Rate) and Precision values. These models were evaluated separately for URL and JavaScript based features. The experiment gave better results

for the J48 Random Forest with better FPR (False Positive Rate) values. However, the research does not explain the performance evaluation metrics chosen to evaluate the experiment.

Automatic cross site scripting (XSS) detection experiment by Nunan et al. (2012), Komiya et al. (2011) used features extracted from the URL and web document content to apply classification techniques to malicious web pages. The feature selection in this research is varied and vast ranging from obfuscation based, suspicious pattern based and Html/JavaScripts schemes based methods, etc.

The models used in the research were Naive Bayes and SVM (Support Vector Machine) and the performance measures used were the Detection Rate, False Alarm Rate and Accuracy Rate. As per the experiment the SVM (Support Vector Machine) with a polynomial kernel achieved a higher performance. However, the aim of the project was not to just measure the results and evaluate it against the metrics, but also to compare it against another research by Likarish et al. (2009). This research was more focussed on feature or aspect based model comparison and performance. However, the research does not provide assurance of the model being capable of detecting all types of scripting attacks, though automated.

We see multiple models created to classify these malicious web codes and one of them which uses a deep learning framework was developed by Wang et al. (2016). This research was conducted to build a model on deep learning to detect malicious JavaScript code. This paper uses stacked denoising autoencoders (SDA) feature selection method to classify the malicious and benign web codes. This research involved multiple models such as Logistic Regression and SVM (Support Vector Machine) which used the stacked denoising autoencoders (SDA) for feature selection. This type of feature selection was concluded to be better than PCA (Principal Component Analysis), ICA (Independent Component Analysis) and FA (Factor Analysis). This research was on the model SDA-LR which was a neural net with logistic regression which had better statistical output than other models. The issue faced with this research objective and goal was that, the training time of this particular model was more due to the training of neurons at different layers. Not only this, but the drawback is mentioned that the classifier fails to detect or identify the benign scripts from malicious one's effectively.

Research on server side scripting has been done by Kamtuo and Soomlek (2016), describing the use of compiler platform with data mining algorithms to detect SQL (Structured Query Language) injections vulnerability within incorrect queries. The models varied from SVM (Support Vector Machine), Boosted Decision Tree, Artificial Neural Network and Decision Jungle. Here, Decision Jungle performed better than other models. However, this paper deals with server side scripting, and does not yet confirm if it works for the detection and prediction of the SQL (Structured Query Language) injection, requiring future work.

Hence, for the detection and prediction of such type of (SQLIA) Structured Query Language Injection Attacks, we have the research done by Uwagbole et al. (2017). 2017 which deals with the addressing of predictive analytics in the field of (SQLIA) Structured Query Language Injection Attacks, on a larger scale with reference to Big Data Analytics. The

methodology involved text pre-processing, feature hashing, filter-based feature selection for top relevant vectors and then splitting the data to train and test the models, later evaluating them for the results. Shar and Tan (2013) empirically evaluated the model at 98.6% reaching a good accuracy level. However, the scope of multi class classifier was not delved into, in order for us to understand if the same can be applied to different classes, as big data may not involve a separation of just one class.

Dimensionality techniques have long been used in different mathematical complex problems to simplify them. Some of it may be due to the fact that it is highly dimensional or some data being redundant. Even in machine learning algorithms, during preprocessing a dimensionality reduction is applied to make it more apt for the model to get the desired output. A research carried out by Sorzano et al. (2014) had a survey done for the dimensionality reduction techniques. Thereafter, finding a plethora of available techniques but addressing the same problem of reducing complexity. Hence, dimensionality reduction is a concept which is user based but more of data based as well. Since, the user is required to understand the data effectively to understand its dimensions and hence reducing it for better computation performance. Therefore, its a subjective matter. The research is available at length with the available techniques to reduce the dimensionality. By and large an issue dimensionality is referred to a curse of dimensionality rather. Due to the fact that certain dimensions in data are very important to get rid of. Data inconsistencies can be another factor which can be looked into further to check about the data dimensionality.

A simple classification technique was used by Needell et al. (2017) from binary data proposing a framework with reduced resource costs and low framework. They have used a supervised learning algorithm for classification which functions on binary data. However, at this stage only a theoretical use case was available.

However, A research carried out by Landgraf and Lee (2015) proposed a novel idea for the Logistic Principal component analysis (PCA) used for binary data. They extended the Pearsons formulation of a low represented dimensional data to binary data with minimum error. This new method does not require any factorization, but natural parameter projection from the saturated model. Hence, the principal component scores on the new data along with the number of observations are simply computed by matrix multiplication. Therefore, LSVD shows a lower deviance than LPCA.

Research done by Gilbert et al. (2017), Yan et al. (2018) shows Convolutional Neural nets are invertible and shows reasonable constructions. Convolutional networks have been used on image and text classification for quite some time now. Usage of these deep learning algorithms has been useful in many terms as they are more efficient than matrix multiplication, though being mathematically equivalent.

3 Methodology

The methodology used in this research is the CRISP-DM approach. CRISP-DM specifically means Cross Industry Process for Data Mining. It is a structured approach with a proven methodology with being robust and practically powerful. The usefulness and

flexibility helps to take all aspects of data mining into consideration.

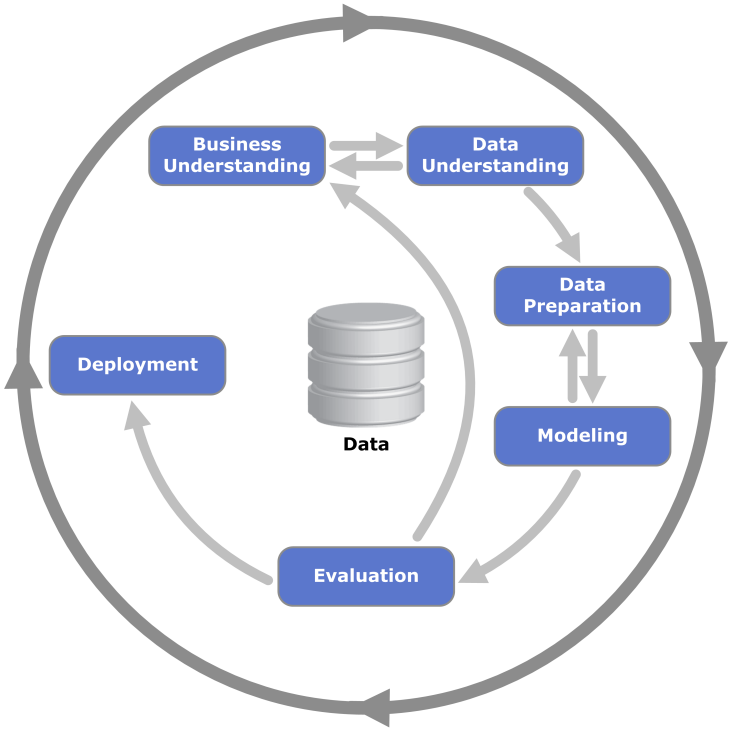

Figure 1: CRISP-DM Cycle Flow

Business understanding In this stage we understood the project requirements and the objectives undertaken. We also considered the feasibility of the project based on the research question posed. We identified the missing factors in the project undertaken, which in our case the objective was to classify malicious and non-malicious web code using new way of feature extraction technique. We then assessed and evaluated the project details and timeline for applying the necessary data mining techniques.

Data understanding - The data was extracted from web pages, where there were two web code scripts. One was the benign script and the other being the malicious. We understood the requirements of the data and the values and how it shall fulfill the project requirements going forward to pre process it and data mine it, to gain the necessary insights and build the required models.

Data preparation- In this stage of the research the cleaning of the data was carried out using the necessary technologies and programming languages using Python, We assess both malicious and non-malicious data in terms of length and other factors and convert from text to binary matrix. During the conversion 0 is appended with the dataset to have consistent length in each row. We also try to eliminate the null or void cells and convert most of the data to the binary format based on the project requirements. As the resulting number of features was high, we also perform a dimension reduction to get the desired dataset which was ready to be trained and modelled accordingly, based on the requirements.

Modeling- In this stage, we split the data set into training and testing, building the

Naive Bayes model first to assess the data performance on how it is getting trained. Going further, additional models like KNN, Random Forest, CNN are trained by tuning the appropriate parameters.

Deployment- In this stage the models are deployed, trying to configure and tune them for the desired results. We also analyse if the models created are suitable, also to check if the data used is helping us with the outlined project objectives and successful deployment.

Evaluation- This is the last stage of the process where the model is being tested and evaluated for the results it produced. We match these results with the project objectives to check if it meets the required standards or if they may require some hyperparameter tuning. We also check and verify if it specifically answers the research question posed in this research project paper, clearly understanding the goal of the project.

4 Implementation

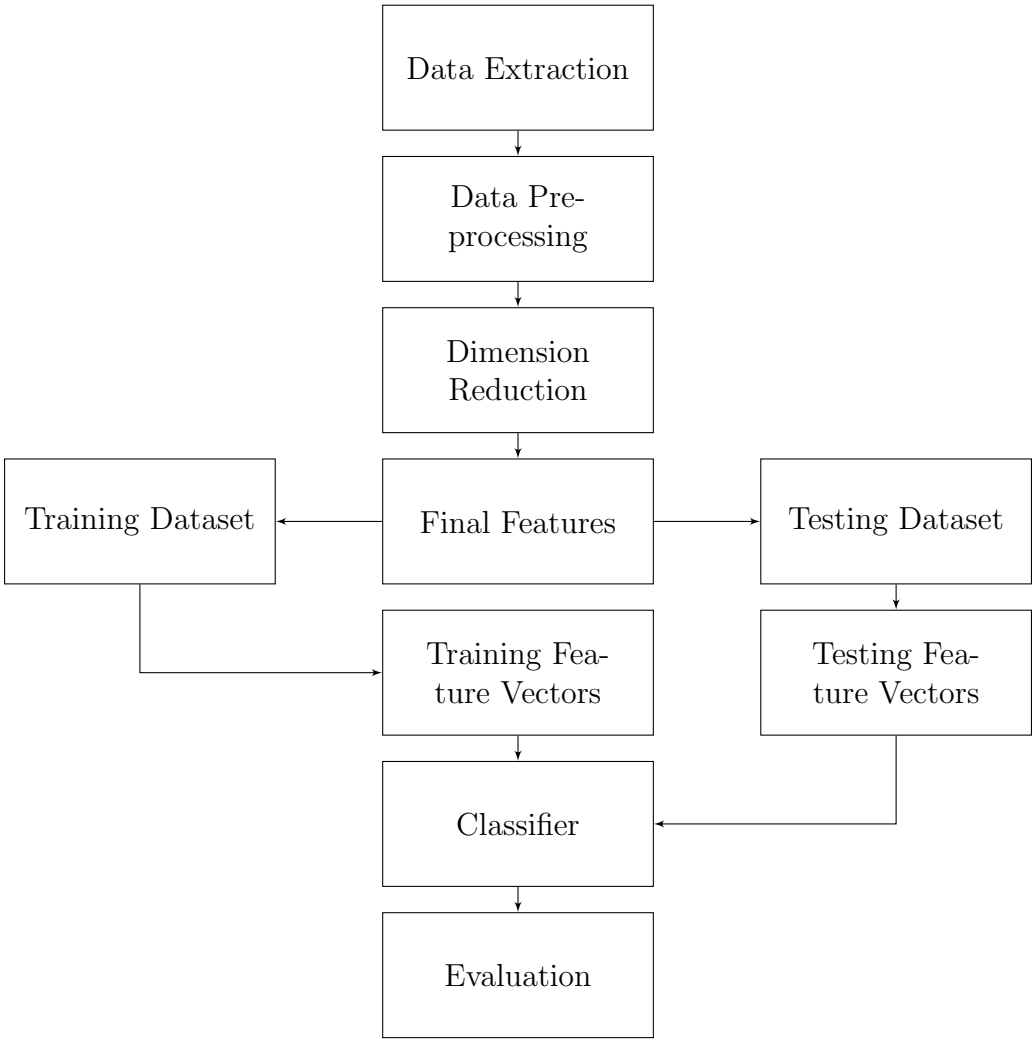

Figure 2: Decision Model

4.1 Dataset

The dataset contains 10500 rows with the combination of both benign and malicious code samples. Each row consist of text samples of different HTML tags which has tag attributes in it. These code samples are scrapped from various websites and stored in two different .csv file as malicious and benign. The composed dataset contains nearly around 5000 rows of malicious code and 5500 rows of benign code samples.

The malicious XSS tag vectors are collected from different websites which are exposed by the hackers for training purpose. These codes contain both regular and encrypted text contents. Whereas, the benign codes were collected from different trusted websites. To ensure that the websites are trusted, a list of top 100 websites list were used, from which code samples from these randomly selected websites were obtained.

Class Level	**Name**	**Training Set**	**Testing Set**
0	Benign Code	3399	1477
1	Malicious Code	4295	1821

4.2 Data Preprocessing

In most cases if there is any textual data NLP technique were used to convert text into meaningful data metrics. In our case the text is composed of HTML tags and attributes in it. In the malicious code samples there are existence of unicode symbols and other meaningless strings. Therefore the dataset is not suitable for any NLP techniques.

The entire dataset is converted into ASCII code matrix to obtain feature vectors. At first, we took a character from the dataset converted into lower case character and generated ASCII value for it. This process is repeated for each and every character in the entire dataset and the resulted ASCII characters are stored in CSV file where each character is separated by a delimiter. The length of the text in each row is different from each which makes the obtained ASCII matrix to have a different length. zeros are appended at the end of each row to have consistent length across the row matrix.

<	———>	0111100	S	———>	1010011
I	———>	1001001	R	———>	1010010
M	———>	1001101	C	———>	1000011
G	———>	1000111	=	———>	0111101

4.3 Dimension Reduction

The processed data will produce 2,902 featuresT the data usually possess a higher number of features. For experiment purpose, we have used dataset of limited text length as the processing time will be higher. Since the length is limited the outcome dataset has less number of features. This 2,902 features are further reduced using dimension reduction technique to the lessen the processing time and to get rid of unwanted features. Different dimension reduction techniques like PCA, ICA, Factor analysis can be used for reducing the dimension. Considering the dataset which is sparse, Numerics Sparse Dimension Reduction technique is used. The resulting reduced dimension dataset will have 446 dimensions. The reduction rate varies depending upon the EPS(Parameter to

control the quality of the embedding) value.

We took a CNN as a baseline classifier and trained the model with original dataset. Further the dimension is reduced using Sparse Dimension Reduction technique. The reduced dimension is used to train the CNN and the accuracy is calculated. The process is repeated using different EPS parameters and accuracy is plotted to check how accuracy is varying based on the number of components used. From the graph we can see that the accuracy gets reduced when the number of components (features) is reduced than 340.

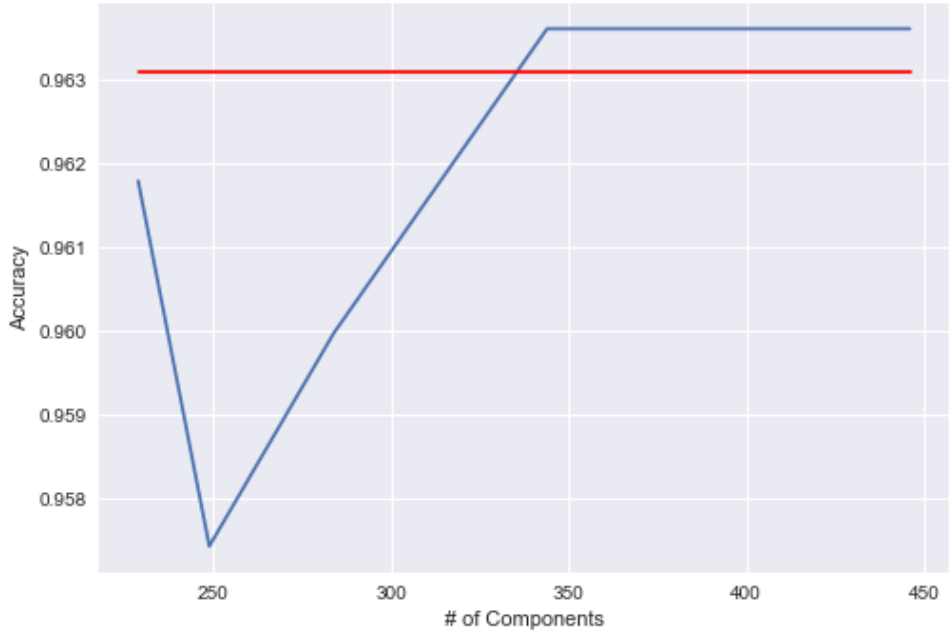

Figure 3: Accuracy of Sparse Projection on Digits

4.4 Classifier

The reduced dimension data is used as an input for various Supervised learning algorithms. We have used Deep learning algorithm as our major classifier. Other classifiers like KNN, Random Forest, Naive Bayes classifier were also used to compare the performance of the model. Considering the features which were extracted from text, we have used Convolution Neural Network as our deep learning algorithm for classification. CNN is capable of handling large number of features and mainly the feature selection is done automatically by the algorithm itself. CNN use multiples of convolution layers to identify the patterns from the test and training dataset.

The reason for choosing CNN among other deep learning classifier is because of its design. The CNN performs good for image recognition where the image pixels are converted to numerics and fed as input. It slides through the features to form different layer of filters. Further these filters are used to understand the pattern in it. The images are two-dimensional whereas the text is one-dimensional. Similarly in our case, the text is converted into numeric values which is fed as an input to the CNN. The CNN slides through the numeric vector to understand the pattern and forms multiple layer of filters.

These filters are further used for classification. Relu and Softmax functions are used as activation function.

4.5 Experiment

For experiment purpose, 70% of the dataset is used as training dataset and 30% is used as testing dataset. The training dataset is fed as input to different classifiers to train the model. Appropriate tuning parameters were chosen for generating our model by using different evaluation technique. Each algorithm has its own tuning parameter. For KNN there are three different K-Value which are used based on the dimensions of the dataset. For CNN different number of hidden layers are used ranging from 0 to 5, wherein for each iteration the values are changed. As the dataset is one dimensional we have used 1D convolution neural network. The epoch value is also an important parameter for model to achieve higher accuracy. If the epoch value is too small, the accuracy will be affected. For training the model we have used the epoch value to range from 200 to 600 with an increment of 100 for each run. In the CNN, the number of hidden layers has direct impact with the performance of the model. Due to the limitations in system requirements and considering the processing time we have used 3 hidden layers for this experiment, where increasing number of hidden layers has a direct impact with model performance.

5 Evaluation

5.1 Modelling Results

The experiment was conducted to mainly focus on deep learning models to better understand the performance in which various other classification algorithms were also used. The entire dataset is divided into 10 different folds. The value for number of iteration is chosen by considering the number of sample size. For each iteration there is a change of 1000 samples in the training dataset.

Naive bayes algorithm is suitable for any classification problem. Considering the dataset, we have chosen Gaussian Naive Bayes as it performs well with normally distributed data. From the K fold test result we can see there is a minimal change in the accuracy between different folds. And so there is less possibility of model overfitting.

K nearest neighbours is one of the most widely used classification technique. Choosing the optimal K value was a challenge. A random test was carried by increasing the K value to understand the model performance by comparing accuracy, validation and training error rate. Initially the error rate started to decrease wherein after a certain minimum there was a slight change in error rate. When tested with the validation set, the model performed well with higher accuracy for K value 10. Random forest is one good algorithm for classification problems, where number of hierarchical decision trees classifiers are constructed by using various sub samples of dataset for prediction. Since its a classification problem, we are using AUC (Area Under Curve) as evaluation metric. Number of trees and tree depth are the tuning parameter for decision tree. To find legitimate N value, random series of values are used. Larger N value leads to higher number of trees which slows downs the processing speed. For choosing the depth value, different decision tree is fit with different depth value ranging from 1 to 30. We plotted the test and training errors to check the overfitting of model.

Naive Bayes			
CV	**Accuracy**	**Precision**	**Recall**
Fold 1	0.459091	0.728311	0.504167
Fold 2	0.429091	0.712717	0.505512
Fold 3	0.4404	0.718148	0.506421
Fold 4	0.453139	0.724817	0.505757
Fold 5	0.454049	0.725526	0.50495
Fold 6	0.43949	0.718978	0.502423
Fold 7	0.463148	0.730594	0.503367
Fold 8	0.4495	0.723492	0.504098
Fold 9	0.459509	0.728022	0.505824
Fold 10	0.44768	0.720534	0.510484
Average	**0.4495097**	**0.7231139**	**0.5053003**

Table 1: K Fold CV performance metrics - Naive Bayes

KNN			
	Accuracy	**Precision**	**Recall**
Fold 1	0.965455	0.965513	0.964956
Fold 2	0.967273	0.967099	0.966743
Fold 3	0.965423	0.965557	0.96429
Fold 4	0.969973	0.969602	0.969383
Fold 5	0.968153	0.967098	0.9682
Fold 6	0.972702	0.972429	0.972684
Fold 7	0.968153	0.967675	0.967863
Fold 8	0.963603	0.961814	0.964669
Fold 9	0.963603	0.963156	0.962002
Fold 10	0.964513	0.965425	0.96308
Average	**0.9668851**	**0.9665368**	**0.966387**

Table 2: K Fold CV performance metrics - KNN

Random Forest			
CV	**Accuracy**	**Precision**	**Recall**
Fold 1	0.966364	0.967414	0.965153
Fold 2	0.96	0.961319	0.957553
Fold 3	0.974522	0.976115	0.972511
Fold 4	0.966333	0.967062	0.964965
Fold 5	0.981802	0.98239	0.980932
Fold 6	0.959964	0.962278	0.956624
Fold 7	0.970883	0.972191	0.968695
Fold 8	0.967243	0.967371	0.965983
Fold 9	0.965423	0.968003	0.96262
Fold 10	0.969063	0.969243	0.967908
Average	**0.9681597**	**0.9693386**	**0.9662944**

Table 3: K Fold CV performance metrics - Random Forest

CNN provides slightly high accuracy when compared to other models. The model performed well using 4 hidden layers and the accuracy of different folds has a little change in the value when validated with testing dataset having less error rate which shows that the model performs well.

CNN			
CV	**Accuracy**	**Precision**	**Recall**
Fold 1	0.982727	0.984397	0.980815
Fold 2	0.983636	0.984061	0.982837
Fold 3	0.984531	0.986422	0.982653
Fold 4	0.981802	0.983588	0.980129
Fold 5	0.984531	0.986462	0.982219
Fold 6	0.988171	0.988508	0.987849
Fold 7	0.988171	0.989617	0.986626
Fold 8	0.981802	0.98881	0.981057
Fold 9	0.989991	0.98219	0.988518
Fold 10	0.988171	0.990867	0.987248
Average	**0.9853533**	**0.9864922**	**0.9839951**

Table 4: K Fold CV performance metrics - CNN

To test the exact performance of the model we have used a separate set of validation dataset which contains new malicious and benign code. The results of the test performed are given in confusion matrix.

	KNN		**Random Forest**		**Naive Bayes**	
	Malicious	**Benign**	**Malicious**	**Benign**	**Malicious**	**Benign**
Malicious	2946	92	3046	34	37	0
Benign	83	2375	29	2387	3023	2436

Table 5: Confusion Matrix

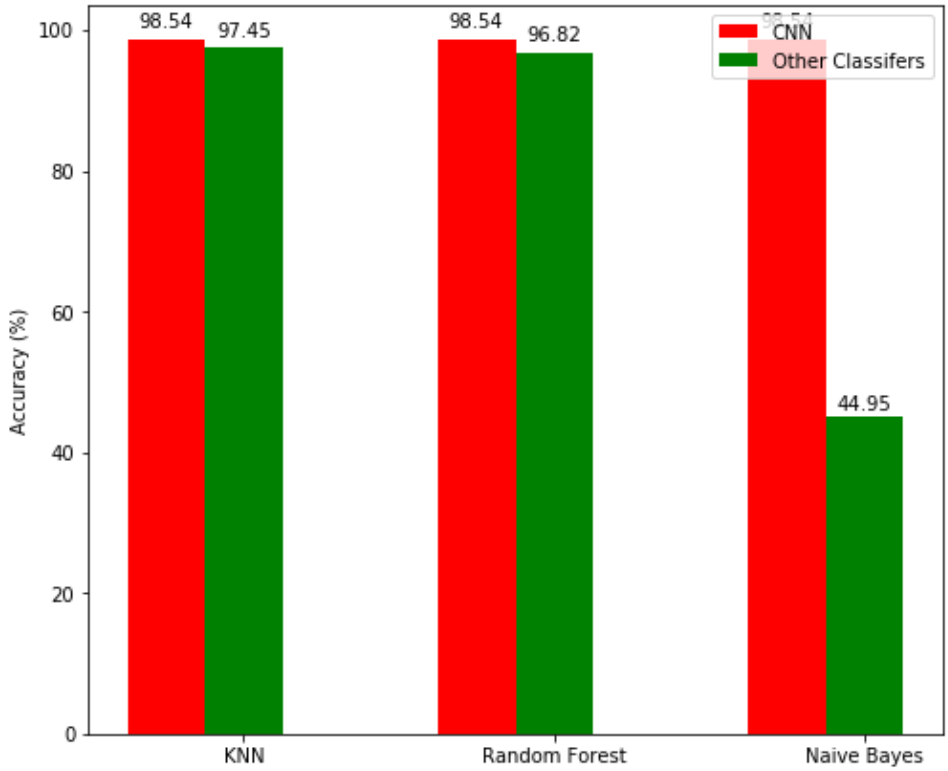

Figure 4: Performance Comparison of models

The evaluation of models is presented by a bar graph which compares and plots the percentage accuracy of Convolutional Neural Networks(CNN) with other models like KNN, Random Forest and Naive Bayes. The Data is represented in two colours where red colour stands for CNN and green colour stands for other models as shown in the graph key. Here, we can see that CNN (98.54%) achieves significantly superior results compared to Naive Bayes(44.95%). We can see a trivial enhancement to the accuracy of the KNN(97.54%) and Random Forest Model(96.82%). Hence, CNN proves to be a better contender among the other models for the detection of malicious code.

6 Conclusion and Future Work

This paper demonstrated the new approach of feature extraction for processing the web code to classify malicious and non malicious webcode. The extracted feature was used to train different classifier where the CNN performed well with the accuracy of 98.5% when compared to other classifier like KNN and Random Forest. This proves that the new technique proposed, which was an amalgamation of feature extraction method and convolutional neural network gives state of the art results.

The accuracy resulted from the evaluation of this model can degrade if used with a large dataset. We consider this limitation for the future work concerning the present research objective. Hence, a novel technique of converting textual data to numeric data for feature extraction and using convolutional neural network to classify the malicious web code from the non-malicious web code proves to be a successful change to the conventional methods.

References

Fawaz, A. M. and Jacob, M. H. (2018). *Detecting Cross-Site Scripting Attacks Using Machine Learning*, Springer Berlin Heidelberg, New York, NY.

Ghaffarian, S. M. and Shahriari, H. R. (2017). Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey, *ACM Computing Surveys* **50**(4): 1–36.
URL: *http://dl.acm.org/citation.cfm?doid=3135069.3092566*

Gilbert, A. C., Zhang, Y., Lee, K., Zhang, Y. and Lee, H. (2017). Towards understanding the invertibility of convolutional neural networks, *arXiv preprint arXiv:1705.08664* .

Gupta, M. K., Govil, M. C. and Singh, G. (2015). Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications, *Computer Science and Software Engineering (JCSSE), 2015 12th International Joint Conference on*, IEEE, pp. 162–167.

Kamtuo, K. and Soomlek, C. (2016). Machine Learning for SQL injection prevention on server-side scripting, *Computer Science and Engineering Conference (ICSEC), 2016 International*, IEEE, pp. 1–6.

Komiya, R., Paik, I. and Hisada, M. (2011). Classification of malicious web code by machine learning, *Awareness Science and Technology (iCAST), 2011 3rd International Conference on*, IEEE, pp. 406–411.

Landgraf, A. J. and Lee, Y. (2015). Dimensionality reduction for binary data through the projection of natural parameters, *arXiv preprint arXiv:1510.06112* .

Likarish, P., Jung, E. and Jo, I. (2009). Obfuscated malicious javascript detection using classification techniques, *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, IEEE, pp. 47–54.

Malviya, V. K., Saurav, S. and Gupta, A. (2013). On Security Issues in Web Applications through Cross Site Scripting (XSS), IEEE, pp. 583–588.
URL: *http://ieeexplore.ieee.org/document/6805456/*

Needell, D., Saab, R. and Woolf, T. (2017). Simple classification using binary data, *arXiv preprint arXiv:1707.01945* .

Nunan, A. E., Souto, E., dos Santos, E. M. and Feitosa, E. (2012). Automatic classification of cross-site scripting in web pages using document-based and URL-based features, *Computers and Communications (ISCC), 2012 IEEE Symposium on*, IEEE, pp. 000702–000707.

Rathore, S., Sharma, P. K. and Park, J. H. (2017). XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs, *Journal of Information Processing Systems* .

Shar, L. K. and Tan, H. B. K. (2013). Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns, *Information and Software Technology* **55**(10): 1767–1780.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S0950584913000852*

Sorzano, C. O. S., Vargas, J. and Montano, A. P. (2014). A survey of dimensionality reduction techniques, *arXiv preprint arXiv:1403.2877* .

Uwagbole, S. O., Buchanan, W. J. and Fan, L. (2017). Applied machine learning predictive analytics to SQL injection attack detection and prevention, *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, IEEE, pp. 1087–1090.

Vishnu, B. A. and Jevitha, P. K. (2014). Prediction of Cross-Site Scripting Attack Using Machine Learning Algorithms, ACM Press, pp. 1–5.
URL: *http://dl.acm.org/citation.cfm?doid=2660859.2660969*

Wang, Y., Cai, W.-d. and Wei, P.-c. (2016). A deep learning approach for detecting malicious JavaScript code: Using a deep learning approach to detect JavaScript-based attacks, *Security and Communication Networks* **9**(11): 1520–1534.
URL: *http://doi.wiley.com/10.1002/sec.1441*

Yan, R., Xiao, X., Hu, G., Peng, S. and Jiang, Y. (2018). New deep learning method to detect code injection attacks on hybrid applications, *Journal of Systems and Software* **137**: 67–77.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S0164121217302571*