Analysis of Intrusion Detection Using KNN and SVM Classifiers

NusrathUnnisa A[1], Dr. Manjula Yerva[2], Dr. M. Z. Kurian[3]

Publication
Tumbe
Group of International Journals
A Peer Reviewed Multidisciplinary Journal

Volume - 5 ; Issue – 2
May - August : 2022
ISSN : 2581-8511
Pages : 5 - 12
Article ID : TUMBE050202
Date of Publication: 01/06/2022

Author/s

[1]PG Student, Dept. of ECE.,
Sri Siddhartha Institute of Technology,
Tumakuru, Karnataka, India,
nussukhanum@gmail.com

[2]Assistant Professor, Dept. of ECE.,
Sri Siddhartha Institute of Technology,
Tumakuru Karnataka, India
,manjulayerva@ssit.edu.in

[3]Head of the Department, Dept. of ECE.,
Sri Siddhartha Institute of Technology,
Tumakuru Karnataka, India,
mzkurianvc@yahoo.com

Abstract

Information and communication technology (ICT) plays a vital role in this digital era and so network security has become more important. Exchange of data or of information is quite common via internet and hence security of data is very crucial. Threats to the network are increasing everyday due to the usage of ICT. In order to deal with all these problems we rely on Intrusion detection system which acts as a heart of the network structure whose work is to keep on monitoring the network activities by regularly verifying the connection pattern and flow of packets through that network. In this paper we are proposing a method using KNN(K-nearest neighbour)and SVM (Support Vector machines) classifiers which acts a model for Intrusion detection system (IDS) to classify the attacks as Benign or Malignant. Experimental analysis is done on-Cross-site scripting(XSS) database to judge the implementation of model.

Keywords: Cross-site scripting, Intrusion Detection system, K nearest neighbour classifier, Support Vector machine Classifier, XSS database.

1. Introduction

Many security approaches exist for a system to communicate which can be defective due to poor configuration or software programs, and hence attacks on the computer network always exists. To find an attack on the network an intrusion detection system plays a crucial role which exploits these faults in computer system [1]. An Ideal IDS can be found when it has a detection rate of 100% and gives false positive rate of 0% by involving minor overhead and small level of monitoring.

In general, two approaches are used for intrusion detection such as misuse detection and Anomaly detection. Misuse detection relies on signature database which works by comparing the sample of signatures with those in the database. The misuse detection is more beneficial since it is helpful in finding known intrusion types but they are unable to find novel attacks. On the other hand anomaly detection is used to find the activities which deviate from standard pattern for network systems or users.

To capture the generally used patterns machine learning algorithms have been used which are capable of classifying the new behaviour as either normal or intruded[2]. Anomaly detection systems are capable of finding unknown attacks but they possess elevated false alarm rate when system behaviour and normal user profiles vary widely.

In this paper we are using both KNN and SVM to detect an attack. K nearest neighbour classifier which is a distance based algorithm. This algorithm can be easily implemented by starting with initialising the distance. For example if K =5, we will be directed to the 5 nearest neighbours by calculating Euclidean Distance. This algorithm does not make any assumption about the underlying data since it belongs to non-parametric classification technique. Another machine learning algorithm is Support Vector machine that classify the data based on the points labelled training examples which belongs to one or two classes. The procedure of using SVM is, in the n-dimension feature space to differentiate max margin hyperplane.

The work is carried out on Cross-site scripting (XSS) which is one of the main frequently occurring attacks on web applications and is a type of injection attacks that injects malicious code into safe websites. XSS attacks directly target the application's user instead of targeting the application's host itself. The remaining section of this paper is divided as follows. Section 2 deals with background. Related work is explained in section 3. Experiments carried out with XSS data is discussed is section 4 and the outcomes are discussed in section 5. At the end i.e, in section 6 conclusion is presented.

2. Background

[1] Mystification

The main goal of mystification is to change the meaning of the code and make it difficult to know its meaning or to read by using operators to compound terms to give program constructs or by changing the names of variables or functions. Mystification techniques can be

used by both benign and malicious scripts with their intended function respectively. In order to safeguard privacy or intellectual rights Benign Mystification are used while malicious mystification works on impersonating malicious intentions and evading the static inspection checks[3].

As an example we can use an encoded URL which is an example of malicious script mystification such as

%3Cs c r ipt%3E%0D%0Aal e r t%28document .c o oki e%29%3B%0D%0A
%3C%2Fs c r ipt%3E

After deobfuscation we can get the original script as

<script > alert(document.cookie); < /script >

3. Related Work

In order to deal with Cross-site scripting (XSS),number of approaches has been discussed. The most commonly used approach is sanitisation and escaping for web application developer which helps to avoid untrusted content being read as code [4,5]. Second technique involves randomised namespace prefixes which involves mark-up language element making it harder for an attacker to read these elements [6]. In [7], to allow control of communication rules are generated with a web proxy blocking communication with untrusted sites. To avoid most private data sent to a third party, combination of static and dynamic techniques were used, which works by continuously taking care of the data flow in the browser [8].

To detect XSS attacks machine learning techniques[10] were deployed which are very attractive methods because of their ability to adapt themselves to change and variations in malicious scripts[9]. To detect the malicious code and to classify them a machine learning approach is used by Komiya et al[9] in which feature extraction was done based on two methods such as , blank separation and tokenising. In the first method spaces are used to separate he terms in inputs and for each term a count is calculated which is used for the calculation of feature weight. But in this method the calculated feature weight may be incorrect since characters must be used to separate the terms in input script rather than spaces. The second method assumes that in order to detect the features of malicious code it contains tokens and feature weights are calculated based on count of each term.

4. Methodology

4.1 Datasets

Dataset considered here are XSS which contains number of payloads including both malignant and benign. From the developers sites malicious scripts are taken which are available online [11-14]. Two datasets were accumulated from the XSS dataset among which the first

data set is used for training and the second is used for testing.The URL is given as input to the classifier.

4.2 Selecting Features

Feature selection can be done in two ways such as structural features and behavioural features.

Structural Features: It consists of group of alphanumeric characters which can be present in the script. A script may contain few structural features but if attacker is trying to spoil the security of the web application then the usage of these features may be increased in the script which is to conclude whether the script is mystified or not.

Table 1: Structural Features

Features	Terms
Punctuation	`,!,@,#,$,%,^,&,*,<,>,?,/,[,{,],},:,;,”,”,’,’,\,|,-,~,+,_,(,)
Punctuation combination	><’, “><,[],==,&#

Behavioural features: It includes commands and functions in the script, which can be used by the attacker suspiciously and differently from the benign developer. Which means to say the benign developer don’t want to mask the intent of their code, whereas attackers will use a range of commands to create the malicious script. The present work uses ‘0’ as absence of features and ‘1’ as presence of features.

Table 2: Behavioural Features

Features	Description
Ability to Read	How well a script can be read , how many alphabetical characters used
Articles Used	Document, window, iframe, location, This
Actions	Onload, Onerror
Technique	createelement, String.fromCharCode, Search
Tags	DIV, IMG, <script
Attributes	SRC, Href, Cookie
Reserve	Var
Functions	Eval()
Protocol	HTTP
External File	.js file

4.3 Classifier: URL is used as input to the KNN/SVM classifiers, from which features are extracted. Based on the training data the classifier will give the class of the output as Benign or Malignant.

5. Results

[1] Experiments

Experimentation is done with the help of Matlab R2017a, which focuses on SVM and KNN classifier performance using the XSS database and features described in section 4.

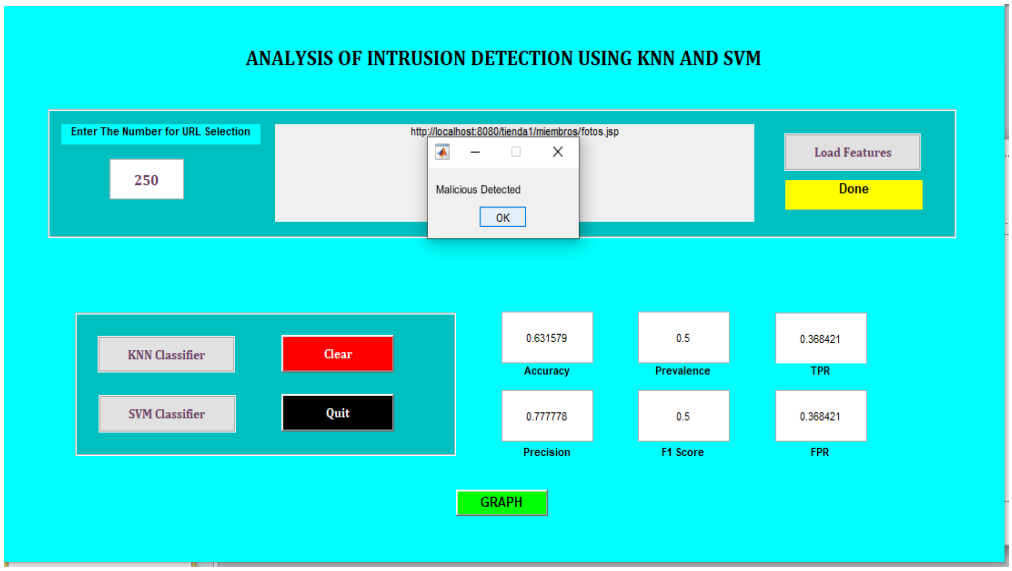

Figure 1: Experimental results using KNN

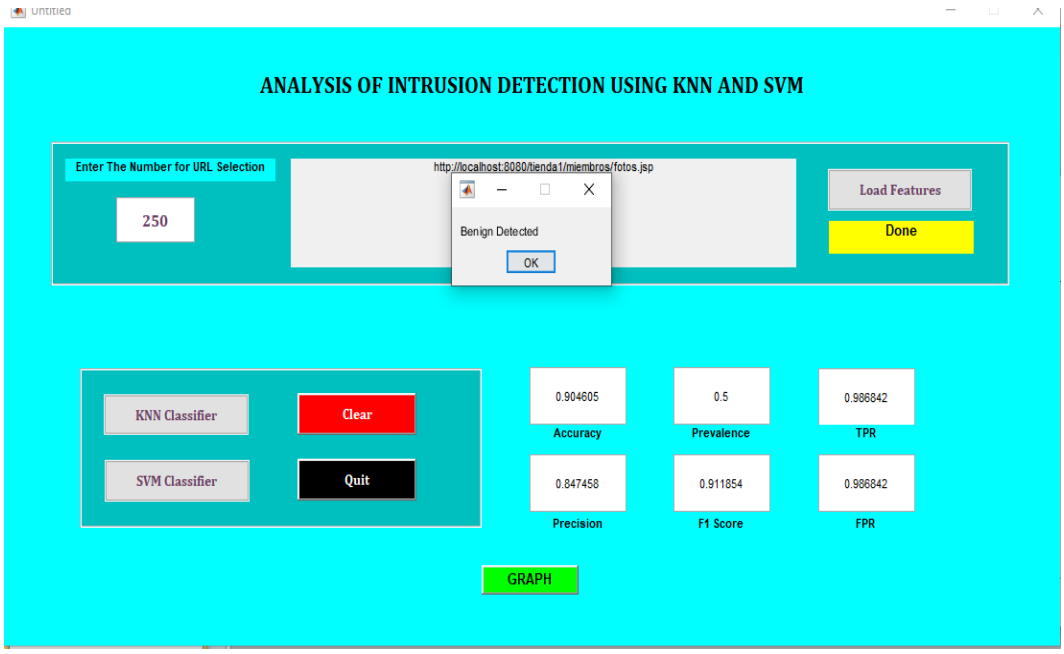

Figure 2: Experimental results using SVM

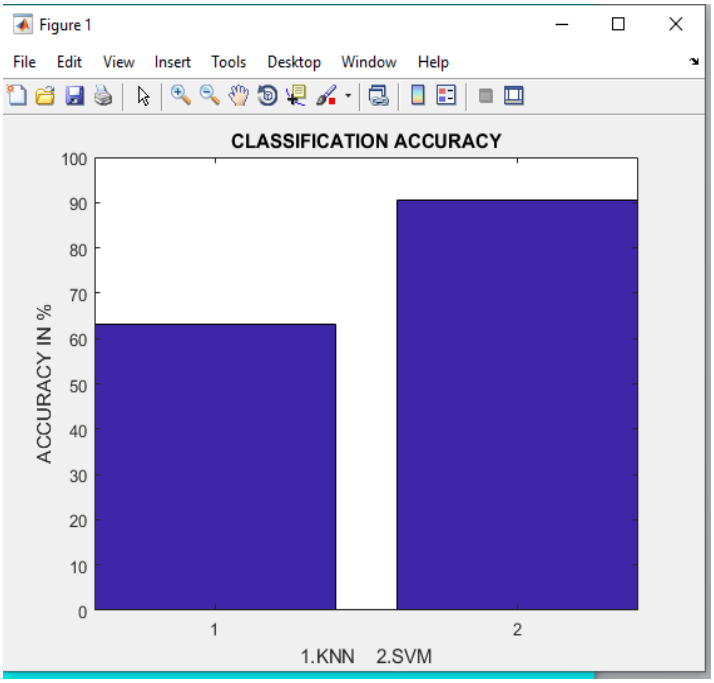

Figure 3: Accuracy Analysis

Table 3: Evaluation of KNN and SVM

	KNN	SVM
Accu-racy	63.15%	90.48%
Preva-lence	50%	50%
TPR	36.84%	98.68%
Precision	77.78%	84.74%
F1 Score	50%	91.18%
FPR	36.84%	98.68%

[2] Discussions

A confusion matrix is generated for the selected URL and the different parameters are calculated such as accuracy, Prevalence, TPR, Precision, F1 score, and FRP.To describe the performance of a classification mode or classifier, a confusion matrix is written on a set of test data for which the true value are known.

Accuracy: To decide how often is the classifier correct in overall. It is given by the formula

$$\frac{TP+TN}{Total}\text{----------(1)}$$

True positive Rate: To decide how often it predict Yes, when it is actually yes. It is calculated as

$$\frac{TP}{Actual\ Yes}\text{----------(2)}$$

False positive Rate: To decide , how often it predicts yes when it is actually no. It is calculated as

$$\frac{FP}{Actual\ No}\text{----------(3)}$$

True Negative Rate: to decide how often it predict no, when its actually no. it can be calculated as,

$$\frac{TN}{Actual\ No}\text{----------(4)}$$

Precision: To decide, how often is it correct when it predicts yes.

$$\frac{TP}{Predicted\ Yes}\text{---------(5)}$$

Prevalence: to decide how often does the yes condition actually occur in our sample. It can be calculated as

$$\frac{Actual\ Yes}{Total}\text{----------(6)}$$

F1 score: It is defined as the harmonic mean of precision and recall.

6. Conclusion

To detect the attacks in XSS coded JavaScript using SVM and KNN classifier is demonstrated in this paper. Based on the training data the classifiers used give the output as Malicious and Benign.Also different parameters are calculated such as accuracy, Prevalence, TPR, Precision, F1 score, and FRP.The classifiers used here can be used as security layers either in a browser or on server to improve the intrusion detection accuracy.

References:

[1] Anderson, J.P. Computer Security Threat Monitoring and Surveillance; Technical Report; James P. Anderson Company: Philadelphia, PA , USA, 1980.

[2] Michie, D.; Spiegelhalter, D.J.; Taylor, C. Machine Learning, Neural and Statistical Classification; Ellis Horwood Series in Artificial Intelligence: New York, NY, USA, 1994; Volume 13.

[3] Xu, W., Zhang, F., Zhu, S.: JStill: mostly static detection of obfuscated malicious JavaScript code. In: Data and Application Security and Privacy, pp. 117–128. ACMPress (2013).

[4] Weinberger, J., Saxena, P., Akhawe, D., Finifter, M., Shin, R., Song, D.: A systematic analysis of XSS sanitization in web application frameworks. In: EuropeanSymposium on Research in Computer Security. Lecture Notes in Computer Science, vol. 6879, pp. 150–171. Springer (2011)

[5] Williams, J., Manico, J., Mattatall, N.: Cross-site Scripting (XSS). https://www.owasp.org/index.php/Cross-site Scripting (XSS).Accessed 22 July 2016.

[6] Van Gundy, M., Chen, H.: Noncespaces: using randomization to defeat cross-site scripting attacks. Comput.Secur. 31(4), 612–628 (2012)

[7] Kirda, E., Kruegel, C., Vigna, G., Jovanovic, N.: Noxes: a client-side solution for mitigating cross-site scripting attacks. In: Symposium on Applied Computing, pp.330–337.ACM Press (2006).

[8] Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., Vigna, G.: Cross site scripting prevention with dynamic data tainting and static analysis. In: Networkand Distributed System Security Symposium, p. 12. Internet Society (2007).

[9] Komiya, R., Paik, I., Hisada, M.: Classification of malicious web code by machine learning. In: Awareness Science & Technology (iCAST), pp. 406–411. IEEE (2011)

[10] Likarish, P., Jung, E., Jo, I.: Obfuscated malicious JavaScript detection using classification techniques. In: Malicious and Unwanted Software (MALWARE), pp. 47-54. IEEE (2009).

[11] Examples of malicious javascript (2014). https://aw-snap.info/articles/js-examples.php. Accessed 16 Dec 2016.

[12] Karnad, K.: XSS payloads you may need as a pen-tester (2014). https://www.linkedin.com/pulse/20140812222156-79939846-xss- vectors-you-may-need-as-a-pen-tester. Accessed 25 Dec 2016.

[13] XSS Payloads: XSS payloads you may need as a pen-tester. http://www.xss-payloads.com/payloads.html. Accessed 14 Oct 2016.

[14] Fernandez, K., Pagkalos, D.: XSS (Cross-Site Scripting) information and vulnerablewebsites archive. XSSed.com. Accessed 14 June 2017. jayant-kaikini-seeping-into-the-surreal/