International Journal of System and Software Engineering
7 (2), December 2019, 15-20
http://www.publishingindia.com/ijsse

# Smart Intrusion Detection Systems Using Machine Learning

Yaseen Khan[1], Zaid Haroon Jan[1] and Nagaveni V.[2*]
[1]UG Student B.E., CSE Department, Acharya Institute of Technology, Bangalore, Karnataka, India.
[2]Professor, CSE Department, Acharya Institute of Technology, Bangalore, Karnataka, India.
Email: nagaveni@acharya.ac.in
*Corresponding Author

**Abstract: Engineering problems are generally solved by gathering domain specific knowledge and then developing algorithms to solve that particular problem. Machine learning is an Engineering discipline that helps us solve these problems by collecting a large datasets and then with the help of that data it is possible to create black box machine output of which can then be used to solve real world problems [1]. This paper cites the basis of machine learning concepts and the methods which can be applied to intrusion detection in the cyber security industry. Machine learning is used to generate predictive models that can be used to classify requests on to a server. Main goal is to build a machine learning model that can classify a request on a server into a normal request or a malicious request based on the server's behaviour and determine the type of attack. Proposing a new system that employs data science community such as normalization through the minmax algorithm, feature selection through the chi square method, planning to separate the data by the Gaussian kernel and then approaching to employ two machine learning models.**

**Keywords: Chi square, Cyber security, Gaussian kernel, Intrusion detection system, Machine learning, Pattern recognition, SVM, Training data.**

## I. Introduction

In today's day and age, where every piece of data we ever "shared" is housed in giant data repositories, the security of this data is of paramount importance. It is important to have our data safe from entities with malicious intents. Methods to protect our data and other systems from attacks and unauthorized access or destruction fall under the broad spectrum of Cyber Security [2].

Firewalls and other conventional methods of Intrusion avoidance have failed to detect the network intrusion without the need for human interference. Smarter ways to deal with intrusions with minimal human interventions should be employed to make the broad sphere of computer networks a safe haven for data and the end users. We employ smart Intrusion Detection Systems (IDS) to deal with the problem described above the main purpose of an intrusion detection system is flag down potential intrusions and preventing them from causing any harm to host system. Intrusion detection systems can be broadly classified into two types:

- *Anomaly Detection*
- *Misuse Detection*

Misuse-based IDS maintain signatures of known intrusion attacks, and network request of known signatures are flagged down. Misuse based intrusion detections system are highly accurate.

However, the flaw with Misuse detection based intrusion detection is that, intrusions manifest themselves in various forms. And these forms crop up on quite a fast rate. Manually maintaining a database for known attack types and their variations is a hard task. The maintenance of these databases can be abstracted away by employing another method of intrusion detection systems called the anomaly based intrusion detection, which work on analysis of the deviation of behaviour from known behaviour types. Due to this fact the anomaly based intrusion detections systems can be used to detect even the novel (Zero-day) attacks. We can even collect data from the anomalous behaviours to create patterns of attack for misuse detection systems.

## II. Research Gap

Many intrusion detection techniques have been proposed. Some based on known types of attacks detection (Misuse Detection) which fail to detect newer attacks and variations of already known attacks. And the fact that the database of known signatures is to be updated frequently poses a problem. Not a lot of research has been made in the use of machine learning techniques and deep learning techniques such as neural networks, decision trees, support vector machines, etc. in cyber security. The paper discusses one of these techniques that can be employed to develop an anomaly detection based intrusion detection system.

## III. Proposed Architecture

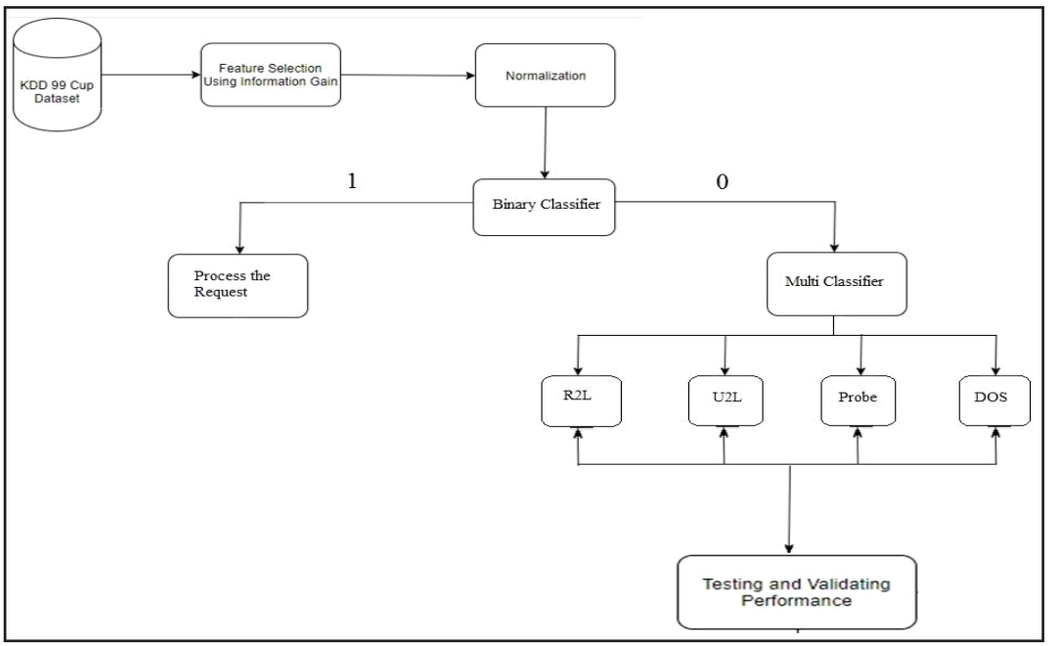


Fig. 1: Proposed System Architecture

## IV. KDD CUP 99 Dataset

The most widely used training datasets in the field of cyber security is 21€ KDD CUP 99 dataset which is based on the DARPA 1998 dataset. These dataset are used for knowledge discovery and data mining tool to build intrusion detector. This database contains a standard set of data to be examined, which includes a variety of intrusions simulated in thousands of network environment. This dataset contains 49,00,000 replicated attacks on record. In this dataset, with the identity of normal there is one normal type and 22 different attack types.

KDD CUP 99 dataset can be broadly classified into five major categories:

- DoS (Denial of Service attacks)
- U2R (User to Root attack)
- Probe (Probing attacks)
- R2L (Root to Local attacks)
- Normal

The dataset contains 41 features attributes along with a target value specifying the type of the networks request denoted by a single record. Out of the 41 attributes, 7 characteristics are of symbolic type, while the rest are continuous. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only. An exhaustive list of types of attacks present in the data set is given below in Table I.

Table I: Types of Attacks

| Classification Identification | Type Identify | Meaning |
|---|---|---|
| Teardrop, Neptune, back, smurf, pod, land, | DOS | Denial of service attacks. |
| Guess_passwd, Imap, multihop, ftp_write, Warezclient, phf,spy, warezmaster | U2R | Unauthorized access from remote machine. |
| Portsweep, Ipsweep, satan, nmap, etc. | Probing | Monitoring and other exploration activities. |
| Perl, Loadmodule, rootkit, buffer_overflow | R2L | Unauthorized access to local super user privileges by ordinary users. |
| Normal | Normal | Normal Record. |

The attributes included for each of the records contains header information of TCP/UDP packets, information about the sessions, along with other attributes like the number of failed login attempts, whether root access was obtained in the session, and other service based data [5].

### *A. Normalization*

The main aim of normalization is to convert the numeric values of the columns in dataset to a common scale normally between 0-1, without altering the differences in the ranges of values.

This is done in order to lower the influence the attribute having values in high orders compared lower order attribute values over the machine learning model. To model the data accurately some algorithms require normalization as part of the pre-processing of the data. Here the Normalization Algorithm used is the Min Max algorithm.

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Which calculates the maximum and minimum values for each of the attributes over the entire dataset and brings the particular instances of each attribute in range from 0-1 [8].

### *B. Feature Selection*

Not all of the features available to us in the KDD dataset will be useful in the formulation of network threat classifier. While they might have some weight in the final prediction of the model, but this weight is negligible. As such it is best to find a certain number of attributes in the dataset which actually have some high order weight associated with them in the final model. Feature selection is the process of selecting the features having high order values associated with in the final model while rejecting the attributes with lower order weights. One of the benefits of the lower of the number of features selected in the final model is the faster network packet evaluation time, since now well be considering fewer variables for the prediction by the model [6].

### *C. Chi-Square Feature Selection (chi)*

Chi-square test is used to evaluate the dependency of an input variable to the output variable. If the dependency to the output variable is significant then the input variable is retained in the dataset. It works on the premise that an output variable is significantly dependent on the input variable if the frequency of change in the output variable is proportional to the frequency of change in the input variable.

Chi-square score is given by:

$$\chi^2 = \frac{(Observed\ frequency - Expected\ frequency)^2}{Expected\ frequency}$$

### *D. Support Vector Machine*

A SVM creates a hyper plane around the instance of a particular class of data. The periphery of this decision hyper plane is called the decision boundary and everything within the confines of the decision boundary is classified to be of a particular class, otherwise not. SVMs can be used directly onto linearly separable data however, for nonlinear data, we employ appropriate kernel functions to map them into higher dimensions so that they become separable in these regions [7].

### *E. SVM Cost Function*

The idea is to train a machine learning model on normal behaviour of a system using labelled data for our model. The decision boundary will be trained using algorithm given by the equation:

$$\min_{\theta} C \sum_{1}^{m} (y^{(i)} cost_1(\theta^T x^{(i)}) + 1 - y^{(i)} cost_0(\theta^T x^{(i)})) + \frac{1}{2} \sum_{1}^{m} \theta_j^2$$

Where C is the regularization parameter,

m is the number of training examples,

θ is a vector of parameter,

x is the input variable,

y is the output variable.

### *F. Gaussian Kernel*

The purpose of a kernel is to make data linearly separable. The aim of every classifier is to predict the classes correctly. For that, the dataset should be separable. It is extremely rare to have a dataset that in which, the data is separable. It usually gives machine learning and deep learning models a hard time. The data can be made linearly separable by transforming an instance to a dataset say x1 to x2. This transformation of data from x1 to x2 to make data separable will be handled by the Gaussian kernel in our model which is given as:

$$f_i = exp(\frac{-\sum_{j=1}^{m}((x_i - l_j^{(i)})^2)}{2\sigma^2})$$

## V. Severity of Threat

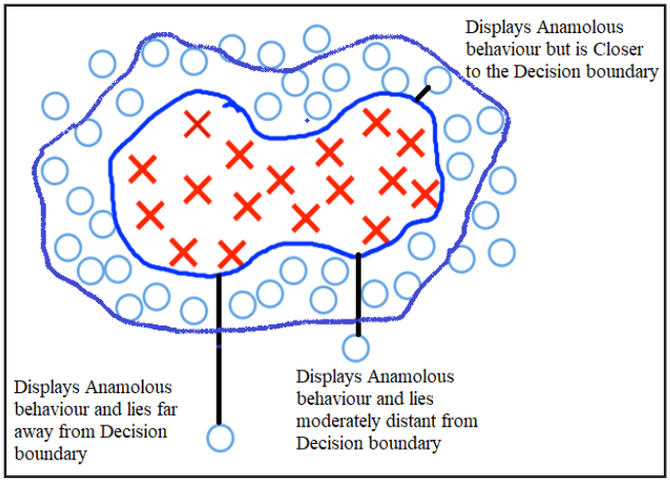


Fig. 2: Threat Severity

Degree of deviation from the Decision Boundary can be used to determine the severity of a threat.

If a point in the hyperspace lies far away and outside the decision boundary then the request can be considered highly malicious and denied service from the Server.

If a point in the hyperspaces lies moderately away and outside the decision boundary then the request can be considered moderately malicious and still denied service from the Server.

If a point lies in the hyperspace lies close to the decision boundary and outside the decision boundary can be considered as a false positive and allowed service from the Server.

Table II: Testing and Validation of Binary Classifier

| | |
|---|---|
| True Negative 78653 | False Positive 637 |
| False Negative 64 | True Positive 19451 |

Total Number of Training Examples Used: 98805

## VI. Results

*Accuracy* = *(True Positive + True Negative)/Total*
= *(78653 + 19451) / 98805*
= *0.9929*
= *99.29 %*

*Precision* = *(True Positive)/(True Positive + False Positive)*
= *19451 / (19451 + 637)*
= *0.9683*
= *96.83%*

*Recall* = *(True Positive)/(True Positive + False negative)*
= *19451 / (19451 + 64)*
= *0.9967*
= *99.67 %*

*F1 Score* = *2 * (Precision * Recall)/(Precision + Recall)*
= *2 * (0.9683 * 0.9967) / (0.9683 + 0.9967)*
= *0.9822*
= *98.22%*

*MCC* = *0.978*

### *A. Multiclass Classifier*

In our model we first run only the binary classifier. If the binary classifier classifies the packet on the server to be normal, the packet is processed. And if the Binary Classifier classifies a packet on the server to be a threat it discards that packet from being processed further and sends that packet to the other machine learning model called Multi Classifier. This Multi Classifier is used to identify which type of attack the packet carried, by examining the attributes of the packet.

Using this architecture in the model we can reduce the time complexity and avoid unnecessary calculations that would have been done if only multi classifier would be used in the classification of a threat. As the binary classifier requires less time when compared with that of multi class classifier. Hence the multi classifier is used only if the packet is identified as threat by the binary classifier. Using the multi classifier, in determining the kind of attack the packet carried, the results can be used further to enhance the systems against these kinds of attacks and know the probability of each type of attack. The results of multi classifier can also be used as signatures in Misuse based IDS.

This Multi Classifier as shown in Fig. 3 classifies the threat into four different categories

- DOS
- PROBE
- R2L
- U2R

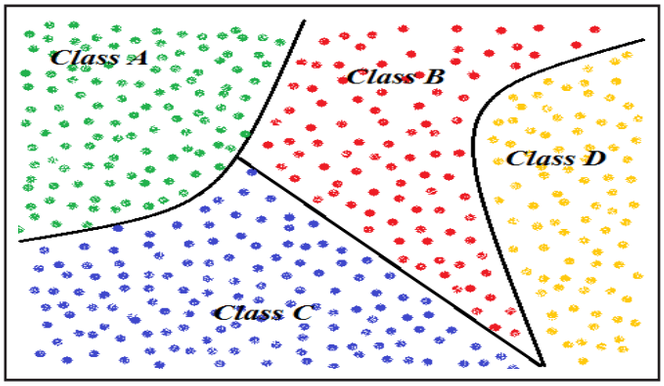


Fig. 3: Multiclass Classifier

Table III: Testing and Validation of Multi Class Classifier

| | DOS | Probe | R2L | U2R |
|---|---|---|---|---|
| DOS | 776532 | 50 | 2 | 0 |
| Probe | 89 | 8200 | 0 | 0 |
| R2L | 18 | 156 | 75 | 0 |
| U2R | 3 | 5 | 0 | 0 |

Table IV: Precision, Recall, and F1 Score of Multi Class Classifier

| | Precision | Recall | F1 Score | Support |
|---|---|---|---|---|
| DOS | 100% | 100% | 100% | 776584 |
| Probe | 97% | 99% | 98% | 8289 |
| R2l | 97% | 30% | 46% | 249 |
| U2R | 0% | 0% | 0% | 0% |

### *B. Pseudo Code*

```
#Convert the symbolic values in the dataset to numeric values
#Scale the features in the dataset in such that one attributes values don't dominate the values of other #attributes
newValue= (oldValue - minValueInPaticularAttribute)/(maxValueInPaticularAttribute - minValueInPaticularAttribute)

#Split the dataset into attributes and label
X = attributes
y = label

#Select 20 features that are most significant to the result of the variable based on the chi-square score of each attribute
selector = SelectKBest(chi2, k=20)
new_data = selector.fit_transform(X, y)

#Split the dataset into training set and test set into X_train, X_test, y_train, y_test such that x_test, y_test #contain 20% of the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

#Configure The SVM classfier
svclassifier = SVC(kernel='rbf', gamma='scale', verbose=1)

#Train The classifier on X_train, Y_train
svclassifier.fit(X_train, y_train)

#Run the classifier on X_test to get a vector with prediction on each test example in x_test
y_pred = svclassifier.predict(X_test)

#Compare the values of y_test, y_pred to get the Confusion Matrix
confusion_matrix(y_test,y_pred)

#Compare the values of y_pred, y_test in order to measure the accuracy, precision, recall, f1 score of the model
classification_report(y_test,y_pred)

#Save the trained model for use on application level
pickle.dump(svclassifier, open('finalizedModel.sav', 'wb'))
```

Fig. 4

### *C. Application of Model*

*Normal Request*

```
F:\Code\Sources\Python\ODU>py application.py
attributes of a request that cause normal behaviour
[[0.0, 0.051282, 0.138297, 0.0, 0.0, 0.0, 0.0, 0.021526, 0.021526, 0.0, 0.0, 0.0, 0.0, 0.094117, 0.623529, 1.0, 0.0, 0.0
4, 0.04, 0.0]]
model loaded
testing testData
Shows less deviation from normal behaviour, can be processed
```

Fig. 5

*Malicious Request*

```
F:\Code\Sources\Python\ODU>py application.py
attributes of a request that causes anomalous behaviour
[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0]]
model loaded
testing testData
Malicious Request, of threat type DOS with threat ranking of Red
```

Fig. 6

## VII. Conclusion

The proposed intrusion detection system integrate multiple techniques which are well known in the data science community such as normalization through the minmax algorithm, feature selection through the chi square method, the data was made separable by the Gaussian kernel and then two machine learning models were employed, where the first model detect whether a network request is malicious or not. And the other one is used to find the type of threat. The architecture of the model is fast and flexible. Classification of network requests into one or the other label takes up time in order of milliseconds, and

the attributes for the prediction in the machine learning model can be maintained with little server overhead, which makes the structure applicable in real time scenarios as a viable intrusion detection and prevention system.

## References

[1] O. Simone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, December 2018.

[2] Y. Xin, L. Kong, ….., and C. Wang, "Machine learning and deep learning methods for cyber security," *IEEE Access*, vol. 6, pp. 35365-35381, May 2018.

[3] H. Saxenam, and V. Richariya, "Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information gain," *International Journal of Computer Applications*, vol. 98, no. 6, July 2014.

[4] B. B. Rao, and K. Swathi, "Fast KNN classifiers for network intrusion detection system," *Indian Journal of Science and Technology*, vol. 10, no. 14, April 2017.

[5] Y. Wang, K. Yang, X. Jing, and H. L. Jin, "Problems of KDD Cup 99 dataset existed and data preprocessing," *Applied Mechanics and Materials*, vol. 667, pp. 218-225, October 2014.

[6] H. Maheta, and K. P. Shroff, "A comparative study of various feature selection techniques in high-dimensional data set to improve classification accuracy," *International Conference on Computer Communication and Informatics (ICCCI), 2015,* India, January 2015.

[7] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, July-August 1998.

[8] P. Trebuna, J. Halcinova, M. Fil'o, and J. Markovic, "The importance of normalization and standardization in the process of clustering," *IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, January 2014.

[9] B. Alic, L. G. Pokvic, and A. Badnjevic, "Machine learning techniques for classification of diabetes and cardiovascular diseases," *6th Mediterranean Conference on Embedded Computing (MECO)*, 2017.

[10] P. H. Swain, and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Transactions on Geoscience Electronics*, vol. 15, no. 3, July 1977.