NETWORK INTRUSION DETECTION SYSTEM

[1]CHUNDURI LALITH CHANDINATH, [2]KANNA PREM CHAND, [3]Kurra Naga Nandini, [4]Shaik Rasul, [5]PRUDHVI GANESH BODDU

***Abstract*: Intrusion detection has become a major concern in the field of network security and administration. Considering intrusion as a security threat, a network needs a system which protects it from known vulnerabilities and unknown vulnerabilities for the efficient functioning of the network. So we are developing an Intrusion detection system which is accurate up to some extent in detecting attacks with a possible minimum number of false positives.**

***Index Terms*: Intrusion, intrusion detection system (IDS), DOS, U2R, Probing, R2L, NSL-KDD Dataset**

I. INTRODUCTION

Intrusions are considered as a sequence of steps taken to compromise the integrity, confidentiality, or availability of valuable assets on computer systems. Intruders gain unauthorized access to the resources available on the system. They use all kinds of techniques to gain access to confidential information and manipulate the data available on the system. This can sometimes damage the system and render it worthless. An IDS can be considered to be a blend of software and hardware units that can be used to identify and pinpoint unauthorized experiments to gain access to the network. All the network-related activities can be audited by an IDS which in turn can be used to suspect the traces of invasions within the network. The end goal of an IDS is to trigger alerts when suspicious activity has occurred by notifying the System Administrator.

Intrusion detection techniques can be classified into two types: a. Anomaly Detection: In this kind of detection, the system alerts malicious tasks by identifying deviations i.e how differently are the network activities occurring as compared to regular patterns. b. Misuse Detection: In this kind of system intrusions are detected based on already known patterns i.e. previously occurred malicious activity. This method can be used to identify and pinpoint known attack patterns more accurately. An ideal IDS will monitor all the happenings within the network and then decide whether those happenings are malicious or normal. The decision is based on system availability, confidentiality, and integrity of the information resources.

An Intrusion Detection System works in the following manner: Collecting Data, Selecting Features, Analysing the Data, and the Actions to be performed.

a) Collecting Data: We need to gather reports on the traffic flowing in the network like hosts alive, protocols used and the various forms of traffic flowing.

b) Selecting Features: After collecting a huge amount of data, the next step is to pick all those required features which we want to work upon.

c) Analysing the Data: In this step, the data about the features which are selected data is evaluated to help us determine if the data is unnatural or not.

d) Actions to be Performed: When a malicious attack has taken place the system administrator is alarmed or notified by the IDS.

The details about the type of attack are also provided by the IDS. The IDS closes the unnecessary network ports and processes to further mitigate the attacks from happening.

II. OBJECTIVE:

Intrusion is considered to compromise the integrity, confidentiality, or availability of valuable assets on computer systems. An intrusion detection system (IDS) audits the traffic flowing in the network for suspicious activity. It then alerts the system administrator when any malicious activity is discovered in the network. The primary functions of intrusion detection systems are discovering anomalies and producing detailed reports for the intrusions discovered. An IDS is programmable software that is developed to detect intrusions within the network. It is employed to hunt and pinpoint the intruders causing chaos within the network. The main principles of IDS are integrity, availability, confidentiality, and accountability. An IDS is built using both software and hardware. It can detect highly dangerous intrusions within the network. The main purpose of IDS is to detect unauthorized packets and malicious communications that happen in computer systems and networks. The most vital ingredient for the success of intrusion detection systems is feature selection.

III. LITERATURE SURVEY

S.No	Paper Title	Name of the Conference/Journal	Technology Used
1	Intrusion Detection System using decision tree algorithm	2012 IEEE 14th International Conference on Communication Technology	In this paper, they analyzed a classification model for misuse and anomaly attack
2	Effective Network Intrusion Detection using Classifiers	Researchgate. in January 2010	In this research, machine learning is being investigated

	Decision Trees and Decision Rules		as a technique for making the selection, using as training data and their outcome. In this paper, they evaluate the performance of a set of classifier algorithms of rules.
3	Intrusion Detection System Based on Decision Tree over Big Data in Fog Environment.	Hindawi, Volume 2018 |Article ID 4680867	Our proposed method not only completely detects four kinds of attacks but also enables the detection of twenty-two kinds of attacks.
4	Implementation of network intrusion detection system using a variant of decision tree algorithm.	2015 IEEE 14th International Conference on Communication Technology	C4.5 Decision tree with pruning we have considered only discrete value attributes for classification. We have used the KDDCup'99 and NSL_KDD dataset to train and test the classifier. The Experimental Result shows that, the C4.5 decision tree with the pruning approach is giving better results with all most 98% of accuracy.
5	Intrusion detection System	2017 International Journal of Technical Research and Applications	The main objective of this paper is to provide a complete study about intrusion detection, types of intrusion detection methods, types of attacks, different tools and techniques, research needs, challenges, and finally develop the IDS Tool for Research Purposes.
6	Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach	Procedia Computer Science Volume 48, 2015	This research work proposed a new approach called outlier detection where the anomaly dataset is measured by the Neighborhood Outlier Factor (NOF). Here, the trained model consists of big datasets with distributed storage environment for improving the performance of the Intrusion Detection system.
7	Recent Advancements in Intrusion Detection Systems for the Internet of Things.	Hindwai, Volume 2019 |Article ID 4301409	In this article, an overview about IDSs suited for IoT networks is given.
8	Fast Anomaly Detection in Multi-Aspect Streams	The Web Conference (WWW), 2021	This article defines a streaming multi-aspect data anomaly detection framework, termed MSTREAM which can detect unusual group anomalies as they occur, in a dynamic manner

IV. INFORMATION SECURITY CONCEPTS USED:

The four attack categories available within the NSL-KDD data set which we have taken are :

i.DOS: This kind of attack leads to the draining of the victims' resources and makes it incapable in responding to legitimate requests. This is one of the 4 attack categories.
Ex: syn flooding. The suitable features from the dataset for this attack class are: "serror_rate" and "flag_SF".

ii.U2R(unauthorized access to local root privileges):
In this kind of attack, an attacker tries to obtain root/administrator privileges by taking advantage of some vulnerability within the victim's system. The attacker usually uses a traditional account to login into a victim's system. The suitable attributes from the dataset for this attack class are: "root_shell", "service_http", and "dst_host_same_src_port_rate".

iii.Probing:
This kind of attack involves obtaining sensitive information present in the victim's computer/device. The suitable attributes for this attack class are: "Protocol_type_icmp" and "dst_host_same_src_port_rate".

iv.R2L:
This kind of attack involves unapproved access of the victim's device by gaining root access where he/she can view the data within that device with root privileges and all this is done from a far-off (remote) machine by the attacker. E.g. password brute force attack. The suitable features from the dataset for this attack class are: "dst_bytes", "dst_host_srv_diff_host_rate", and "dst_host_same_src_port_rate".

V. ARCHITECTURE:

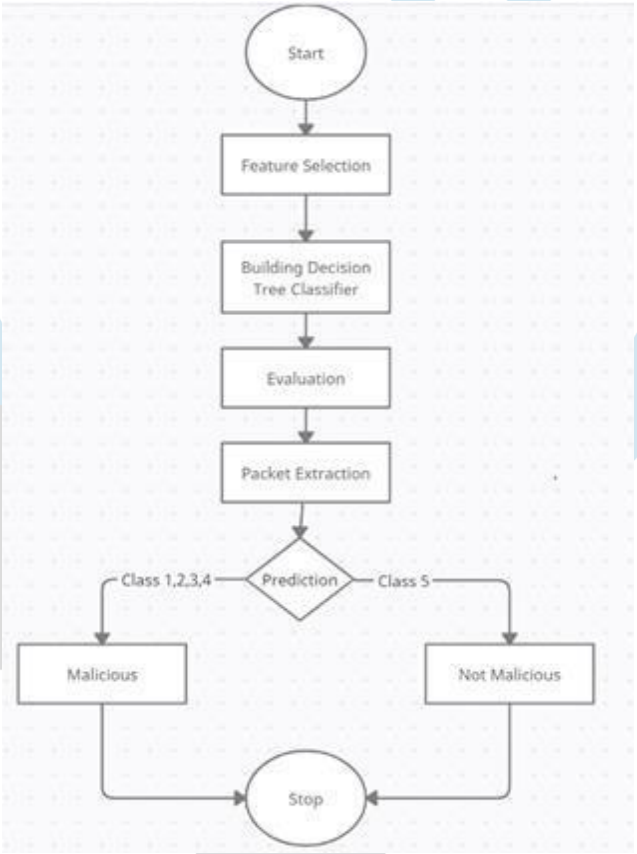

Fig 5.1 Architecture

VI. METHODOLOGY:

The primary goal is to design a plan for detecting intrusions within the system with the least possible number of features within the dataset. Based on the data from previous papers published, we can tell that only a subdivision of features in the dataset are derivative to the Intrusion Detection System. We have to cut back the dimensionality of the dataset to build an improved classifier in a justifiable amount of time.

The approach we are going to use has a total of 4 stages: In the first stage, we pick out the significant features for every class using feature selection. In the next, we combine the various features so that the final cluster of features are optimal and relevant for each attack class. The third stage is for building a classifier. Here, the optimal features found in the previous stage are sent as input into the classifier. In the last stage, we test the model by employing a test dataset.

VII. MODULES:

a. Feature selection:

Here we will be using Information Gain (IG) to select the subset of relevant features in this project. Information Gain often costs less and is faster. We calculate Information gain for all the attributes present in the training dataset. It is calculated for each class separately. In the next step, the values of the information gain are ranked i.e. the feature with the highest information gain being at rank 1. It means that this particular feature can be distinctively classi for the articular clfiedass. If the value of the Information gain is less than the fixed threshold value for a particular feature, that feature can be eliminated from the feature space.

Stage 1: We divide the training dataset into 4 datasets. The training dataset is divided into 4 datasets in such a way that each dataset consists of records belonging to the same attack class along with some of the records of the original dataset. This stage is performed so that the feature selection method is unbiased while selecting features for frequently occurring attacks in the dataset.

Stage 2: In this stage, the datasets for each attack class are sent separately as input into the method used to calculate the information gain. The output of this method gives us the most significant features for each attack type

Stage 3: In the third stage we generate a list of ranked features for each attack class. Now we eliminate all the irrelevant features from the list in accordance with the fixed threshold values.

b. Combining the optimal features:

In this stage, we combine the list of features generated for each attack into a single list. For some of the attack classes the highest ranks i.e. the top 4 features chosen for classification. But for some types of attack classes, we can only take 1 feature since that particular feature is at the top of the rank table and the remaining features are at the very bottom of the table. So, the final set of combined optimal features can be used to entirely distinguish the attack types.

c. Building a classifier:

A Decision tree is an algorithm which takes decisions at each node of the tree and is widely used for regression and classification. It is a supervised learning algorithm in Machine learning where which attribute should be at which node is learned by using a set of labeled examples. The main advantage in using decision trees is that they can be trained very easily and they can even classify nonlinear data. It is more productive than most of the classification algorithms in ML-like K-Nearest Neighbours in most cases. The common measures used to select attributes at each node in Decision trees are Info gain and Gain ratio.

d. Evaluation:

We test the model by employing a test dataset.

VIII. EVALUATIONLUTION AND RESULT:

a) Dos Attack:

```
[61] Y_DoS_pred2=clf_rfeDoS.predict(X_DoS_test2)
     # Create confusion matrix
     pd.crosstab(Y_DoS_test, Y_DoS_pred2, rownames=['Actual attacks'],
                 colnames=['Predicted attacks'])
```

Fig 8.1 Dos Attack Code

Predicted attacks	0	1
Actual attacks		
0	9602	109
1	2625	4835

Fig 8.2 Dos Attack Code Output

b) Probe Attack:

```
[62] Y_Probe_pred2=clf_rfeProbe.predict(X_Probe_test2)
     # Create confusion matrix
     pd.crosstab(Y_Probe_test, Y_Probe_pred2, rownames=['Actual attacks'],
                 colnames=['Predicted attacks'])
```

Fig 8.3 Probe Attack Code

Predicted attacks	0	2
Actual attacks		
0	8709	1002
2	944	1477

Fig 8.4 Probe Attack Code Output

c) R2L Attack:

```
[63] Y_R2L_pred2=clf_rfeR2L.predict(X_R2L_test2)
     # Create confusion matrix
     pd.crosstab(Y_R2L_test, Y_R2L_pred2, rownames=['Actual attacks'],
                 colnames=['Predicted attacks'])
```

Fig 8.5 R2L Attack Code

Predicted attacks	0	3
Actual attacks		
0	9649	62
3	2560	325

Fig 8.6 R2L Attack Code Output

d) U2R Attack:

```
Y_U2R_pred2=clf_rfeU2R.predict(X_U2R_test2)
# Create confusion matrix
pd.crosstab(Y_U2R_test, Y_U2R_pred2, rownames=['Actual attacks'],
            colnames=['Predicted attacks'])
```

Fig 8.7 U2R Attack Code Output

Predicted attacks	0	4
Actual attacks		
0	9706	5
4	52	15

Fig 8.8 U2R Attack Code Output

Cross Validation: Accuracy, Precision, Recall, F-measure.

- Accuracy: The accuracy (AC) is defined as the distribution of the total number of correct predictions. The equation is estimated as:

Accuracy= TP+TN/TP+TN+FP+FN

- Precision: Precision is defined as the ratio of the total no. of true positives and the sum of the number of true positives and the number of false positives. It is calculated by the equation:

Precision 1= TP/TP+FP

- Recall: Recall can be defined as the proportion of the total number of positive examples rightly listed, divided by the total number of positive ones. High Recall suggests correct identification of class. It is also defined in scientific terms as Detection Frequency, True Positive Rate, or Sensitivity. The equation computes as:

Recall = TP/TP+FN

- F-score: The F score is also known as the F1 score or F measure. It is a measure of the accuracy of a test. The F score is interpreted as the weighted harmonic mean of the test's precision and recall. This score is calculated using the following equation:

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precisior}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precisior} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

```
[54] from sklearn.model_selection import cross_val_score
     accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10,
                                scoring='accuracy')
     print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
     precision = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10,
                                 scoring='precision')
     print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
     recall = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10,
                              scoring='recall')
     print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
     f = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10, scoring='f1')
     print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

Fig 8.9 Dos Code

```
Accuracy: 0.99738 (+/- 0.00267)
Precision: 0.99692 (+/- 0.00492)
Recall: 0.99705 (+/- 0.00356)
F-measure: 0.99698 (+/- 0.00307)
```

Fig 8.10 Dos Code Result

```
[59] accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10,
                                scoring='accuracy')
     print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
     precision = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10,
                                 scoring='precision_macro')
     print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
     recall = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10,
                              scoring='recall_macro')
     print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
     f = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10,
                         scoring='f1_macro')
     print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

Fig 8.11 Probe Code

```
Accuracy: 0.99085 (+/- 0.00559)
Precision: 0.98674 (+/- 0.01179)
Recall: 0.98467 (+/- 0.01026)
F-measure: 0.98566 (+/- 0.00871)
```

Fig 8.12 Probe Code Result

```
[56] accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10,
                                scoring='accuracy')
     print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
     precision = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10,
                                 scoring='precision_macro')
     print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
     recall = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10,
                              scoring='recall_macro')
     print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
     f = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10,
                         scoring='f1_macro')
     print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

Fig 8.13 R2L Code

```
Accuracy: 0.97459 (+/- 0.00910)
Precision: 0.96689 (+/- 0.01311)
Recall: 0.96086 (+/- 0.01571)
F-measure: 0.96379 (+/- 0.01305)
```

Fig 8.14 R2L Code Result

```
[57] accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10,
                                scoring='accuracy')
     print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
     precision = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10,
                                 scoring='precision_macro')
     print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
     recall = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10,
                              scoring='recall_macro')
     print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
     f = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10,
                         scoring='f1_macro')
     print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

Fig 8.15 U2R Code

```
Accuracy: 0.99652 (+/- 0.00278)
Precision: 0.87538 (+/- 0.15433)
Recall: 0.89540 (+/- 0.14777)
F-measure: 0.87731 (+/- 0.09647)
```

Fig 8.16 U2R Code Result

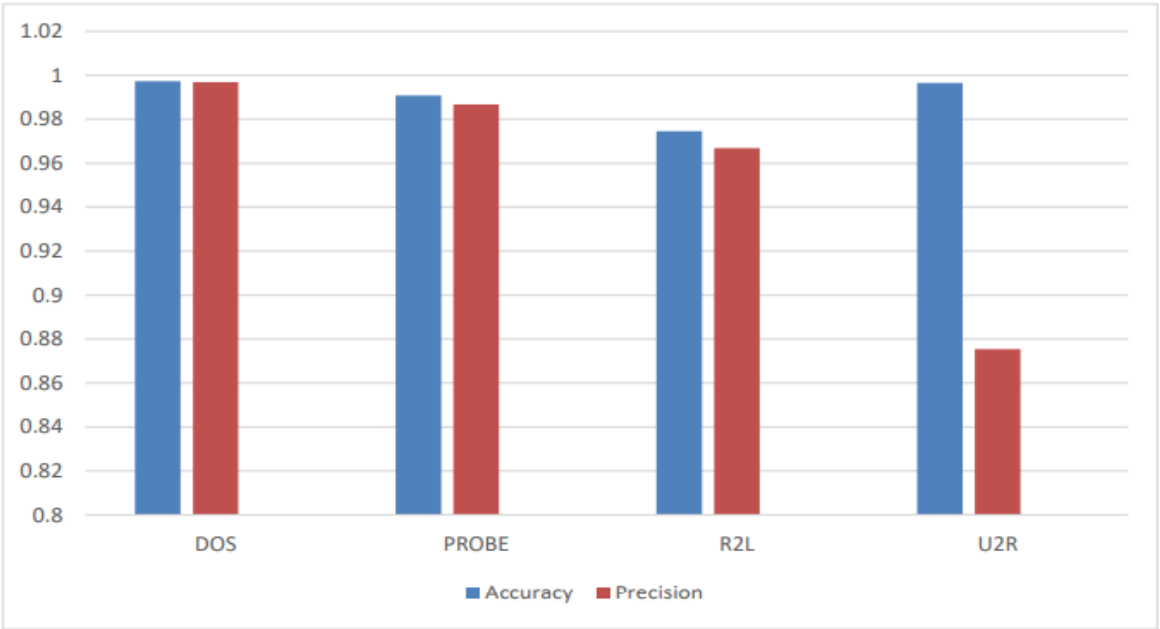

Fig 8.17 Graph for DOS, PROBE, U2R, and R2L against accuracy and precision

IX. CONCLUSION:

Nowadays, almost every system is prone to attacks. There is a need to increase security in our daily use systems. This can be achieved by using Intrusion Detection Systems. It helps us in detecting malicious activities efficiently. Another type of Intrusion detection system is used to detect anomalies. But, the current software which detects anomalies detects a high number of false positives which leads to an increase in the rate of false alarms. Also, the results obtained were highly inaccurate and in many cases, some types of attacks were not able to be detected. Therefore we conducted an experiment to assess the accuracies and detection rates of various algorithms using the NSL-KDD dataset. According to our results, we were able to conclude that our approach i.e. Single Level Multimodel using Decision Trees, has performed exceptionally well and has shown very low rates of false alarms. We have taken into consideration the factors like Accuracy, Precision, Recall, and F-Measure to select our approach.

NSL-KDD Dataset: https://www.kaggle.com/hassan06/nslkdd

REFERENCES

[1] Efficient classification mechanism for network intrusion detection system based on data mining techniques by A. S Subaira; P. Anitha Published in 2014 IEEE 8th International Conference on Intelligent Systems and Control.
[2] Multi-Model Network Intrusion Detection System Using Distributed Feature Extraction and Supervised Learning by Sumeet Dua published in Spring 5-2020.
[3] An Efficient Intrusion Detection System with Convolutional Neural Network by V. Maheshwar Reddy, I. Ravi Prakash Reddy, K. Adi Narayana Reddy in springer conference papers 30 april 2020.
[4] System and method for real-time insertion of data into a multi-dimensional database for network intrusion detection and vulnerability assessment by Robert Gleichauf, Steven Shaklin Published in 2001-08-28
[5] Feature Selection and Deep Learning based Approach for Network Intrusion Detection by Jie Ling, Chengzhi Wu Published on April 2019.
[6] Machine Learning Based research for network intrusion detection: A state-of-the-art by Kanubhai K. Patel, Bharat V. Buddhadev Published in June 2014.
[7] https://www.academia.edu/download/30701138/10.1.1.60.4079.pdf
[8] https://ieeexplore.ieee.org/abstract/document/6511281
[9] http://www.iiard.com/index.php/IJCSMT/article/view/737
[10] https://www.researchgate.net/publication/265187130_Effective_Network_Intrusion_Detection_using_Cl
[11] https://www.hindawi.com/journals/wcmc/2018/4680867/
[12] https://ieeexplore.ieee.org/document/7029925
[13] https://www.researchgate.net/publication/316599266_INTRUSION_DETECTION_SYSTEM
[14] https://www.sciencedirect.com/science/article/pii/S1877050915007000
[15] https://www.hindawi.com/journals/scn/2019/4301409/
[16] https://arxiv.org/abs/2009.08451v4