Network Security

Dataset KDD99 Network Intrusion Detection

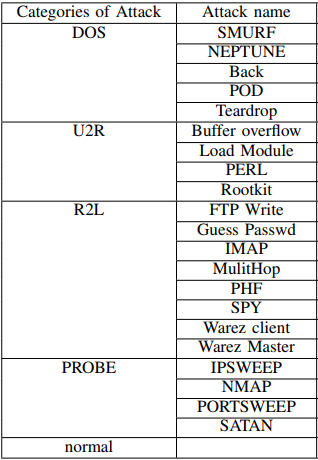


Figure Attack Categories

Figure Attack Categories

Dataset Description

1. Basic features of individual TCP connections.

|  |  |  |
| --- | --- | --- |
| *feature name* | *description* | *type* |
| duration | length (number of seconds) of the connection | continuous |
| protocol\_type | type of the protocol, e.g. tcp, udp, etc. | discrete |
| service | network service on the destination, e.g., http, telnet, etc. | discrete |
| src\_bytes | number of data bytes from source to destination | continuous |
| dst\_bytes | number of data bytes from destination to source | continuous |
| flag | normal or error status of the connection | discrete |
| land | 1 if connection is from/to the same host/port; 0 otherwise | discrete |
| wrong\_fragment | number of ``wrong'' fragments | continuous |
| urgent | number of urgent packets | continuous |

1. Content features within a connection suggested by domain

|  |  |  |
| --- | --- | --- |
| *feature name* | *description* | *type* |
| hot | number of ``hot'' indicators | continuous |
| num\_failed\_logins | number of failed login attempts | continuous |
| logged\_in | 1 if successfully logged in; 0 otherwise | discrete |
| num\_compromised | number of ``compromised'' conditions | continuous |
| root\_shell | 1 if root shell is obtained; 0 otherwise | discrete |
| su\_attempted | 1 if ``su root'' command attempted; 0 otherwise | discrete |
| num\_root | number of ``root'' accesses | continuous |
| num\_file\_creations | number of file creation operations | continuous |
| num\_shells | number of shell prompts | continuous |
| num\_access\_files | number of operations on access control files | continuous |
| num\_outbound\_cmds | number of outbound commands in an ftp session | continuous |
| is\_hot\_login | 1 if the login belongs to the ``hot'' list; 0 otherwise | discrete |
| is\_guest\_login | 1 if the login is a ``guest''login; 0 otherwise | discrete |

1. Traffic features computed using a two-second time window.

|  |  |  |
| --- | --- | --- |
| *feature name* | *description* | *type* |
| count | number of connections to the same host as the current connection in the past two seconds | continuous |
|  | *Note: The following  features refer to these same-host connections.* |  |
| serror\_rate | % of connections that have ``SYN'' errors | continuous |
| rerror\_rate | % of connections that have ``REJ'' errors | continuous |
| same\_srv\_rate | % of connections to the same service | continuous |
| diff\_srv\_rate | % of connections to different services | continuous |
| srv\_count | number of connections to the same service as the current connection in the past two seconds | continuous |
|  | *Note: The following features refer to these same-service connections.* |  |
| srv\_serror\_rate | % of connections that have ``SYN'' errors | continuous |
| srv\_rerror\_rate | % of connections that have ``REJ'' errors | continuous |
| srv\_diff\_host\_rate | % of connections to different hosts | continuous |

Steps:

1. Pre-processing data by removing redundant row values from the dataset.
2. Using Weka to determine accuracy by using correctly classified instances of each of the Machine-learning algorithm.
3. Coding the algorithm with the max precision and accuracy using R.
4. Further deploying that algorithm after parameter tuning for other dataset. (Testing)

Using Weka (Step 2)

**Naïve Bayes:**

Cross Validation (10 Folds)



Percentage Split: (70%)



**Decision Table:**

Cross Validation (10 Folds)



Percentage Split: (70%)



**KNN:**

Cross Validation (10 Folds)



Percentage Split: (70%)



**Random Forest:**

Cross Validation (10 Folds)



Percentage Split: (70%)



**AdaBoostM1**

Cross Validation (10 Folds)



Percentage Split: (70%)



**Attribute Selection:**

Proposed Method

The dataset taken from the Kdd99 is a huge dataset and the one that we have used in our research is under the folder corrected. Our aim is to not only to find the best algorithm suited for the intrusion detection but also to implement it using the programming language R. The first process of applying learning is to pre-process the data. First, we convert the files to CSV format. Then we have to remove the redundant rows from the dataset. Then our next step is to see whether there are any missing values and then to remove those corresponding rows too.

The next process is to use this dataset and put it across various machine-learning algorithms that might give good results by correctly classifying the instances. The tool that we used is the Weka. Weka is an open source Java platform for processing, classifying, clustering and visualization. It is considered as one of the better data mining tools and therefore we have used it. Steps involved in using Weka are

1. Importing the dataset
2. Classifying and choosing the algorithm.
3. Using the *10 Fold* method
4. Again testing using the *Percentage Split* (70%)
5. Checking the *Correctly Classified Instance* Percentage.

*Naïve Bayes* is the first algorithm tested for percentage of correctly classified instances. A probabilistic classification algorithm that has assumptions of high independence between the features. The percentage of correctly classified instances is then noted down.

Cross Validation (10 Folds)



Percentage Split: (70%)



*Decision Table* algorithm was next used and the corresponding output has been shown below.

Cross Validation (10 Folds)



Percentage Split: (70%)



*K Nearest Neighbour* algorithm was next used and the corresponding output has been shown below.

Cross Validation (10 Folds)



Percentage Split: (70%)



Random Forest algorithm was next used and the corresponding output has been shown below.

Cross Validation (10 Folds)



Percentage Split: (70%)



AdaBoost M1algorithm was next used and the corresponding output has been shown below.

Cross Validation (10 Folds)



Percentage Split: (70%)



It can clearly be seen from the above give values that the best algorithm that can be used for the network intrusion detection is the *Random Forest*. Random Forest algorithm is a classification algorithm based on ensemble learning. It works by building multiple decision trees at training and the developed decision trees forms the output function.

The algorithm is elected and now the implementation using the R programming language is to be done. The drawback of this intensive and the accurate algorithm in this case is that the computation time is very high. To lessen the computational time we use *feature selection* algorithm. The feature selection algorithm used is the *InfoGainAttribute* using the Weka tool. Information gain is a feature selection method uses entropy of the class variable and then assess the feature.

Using this method, we expect no considerable drop in accuracy in terms of the percentage of correctly classified classes and a great reduction in time taken to detect intrusion in the network. This makes the system for intrusion detection more efficient.

The attributes elected are used in the R program to predict the error rate and in future to predict if a network is bad or normal. This program when developed fully will act as a filter to determine if a network is secure and will continuously learn from its own series of data making it better and stronger with each type of attack.

**Results:**

The results from the Weka are shown in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cross Validation (10 Folds) | | Percentage Split (70%) | |
|  | Correctly Classified Instances | Incorrectly Classified Instances | Correctly Classified Instances | Incorrectly Classified Instances |
| Naïve Bayes | 58866 (76.16%) | 18425 (23.84%) | 17987 (77.57%) | 5200(22.43%) |
| Decision Table | 76141 (98.51%) | 1150 (1.49%) | 22868 (98.51%) | 319 (1.36%) |
| KNN | 76474 (98.94%) | 817 (1.06%) | 22944 (98.95%) | 243 (1.05%) |
| Random Forest | 76829 (99.40%) | 462 (0.60%) | 23040 (99.37%) | 147 (0.63%) |
| AdaBoost M1 | 68113 (88.13%) | 9178 (11.87%) | 20441 (88.16%) | 2746 (11.84%) |

The Random Forest, KNN and the Decision Table give high accuracy but the accuracy achieved in case of the Random Forest is the highest. Therefore, to further our project we choose the Random Forest algorithm, use it practically to find out the error rate using the R Programming language. Before directly using all the 41 attributes onto the R we do feature selection to select the attributes using the InfoGain method to reduce the time of computation whereas the accuracy is not much compromised. The attributes elected are

* SrcBytes - number of data bytes from source to destination
* DstBytes - number of data bytes from destination to source
* DstHostSameSrvRate – Destination host same server rate.
* Count - number of connections to the same host as the current connection in the past two seconds
* DstHostDiffSrvRate - Destination host different server rate.

The number of attributes reduce from 41 to 5 using the InfoGain method which makes it much faster in terms of execution time.

The R language is then used to see the *OOB* *Estimate of Error* that is retrieved after using the Random Forest machine-learning algorithm. The *OOB* *Estimate of Error* was found to be 1.46% when the half of dataset was used in training and the other half in testing the algorithm. For each of the testing dataset we can predict if the connection setup is normal or malicious and even pin point to the particular type of attack (example – Neptune, smurf, Saturn, teardrop, rootkit etc.) This means that we can be 98.54% sure of the prediction value shown by the algorithm running on R.

Conclusion

For network intrusion, prediction the algorithm that can used is Random Forest as shown by the experiment. It gives a high accuracy rate in determining the classes correctly. The process of attribute or feature selection is a very important as it increases the efficiency of the system as a whole during implementation. The loss in percentage of correctly classified instances due to feature selection is not considerable. The method of using the InfoGain feature selection for dataset with many features can be used when machine-learning algorithm like Random Forest is used.

For future studies, this algorithm can be applied to various other real time original data and its accuracy and be continuously monitored. We expect the fully developed system made using this would be of great success in the real world in predicting if the connection set up is normal or malicious. Based on which the further precautions can be taken for secured network.