Rochester Institute of Technology

Intelligent Security Systems
Project - Intrusion Detection System

Pratik Mehta
Ravina Dandekar
Saunil Desai

Executive Summary

We have implemented Intrusion Detection System using Neural Networks in this Phase. Weka was used in Phase2 to carry out Data Cleaning, Java was previously used in Phase1 to prepare datasets and R was used for building a neural network. We have used the 'neuralnet' package available in R for implementing the neural network. Our implementation consists of multi-layered network that consists of numerous neurons, which are arranged into levels that are interconnected. The nueralnet package used an Input layer, the output layer which will provide the classification and between them there are number of hidden layers. The results obtained show great accuracies and the implementation took a minimal time for completion. Resource consumption was noticeably low too. Data files prepared in Phase1 of the project were used. Dataset was divided into training and testing sets so that we could train the Neural Network and carry out Intrusion Detection.

Specification

Majorly the specification for this project can be listed out as follows

- Implementing and learning how a neural network would perform when detecting an intrusion in the system.
- Testing and learning how a neural network would perform in differentiating the different kinds of attacks from normal behavior.
- Testing and learning the ability of the neural network to distinguish between different types of attacks.
- We design and implement a misuse detection system capable to detect a particular attack where we train our Neural Network on our training set with two output states: Attack of a specified type/ Other Case. After training we test our Neural Network on the testing set.
- We design and implement a misuse detection system that can identify between few different attack types where we train our Neural Network on our training set with outcomes identifying each of the five attacks and then test our Neural Network on the testing set
- We design an anomaly detection system where we train our Neural Network to detect a normal case against any other cases and after training our Neural Network, we test our Neural Network on a testing set.
- We also record time taken and memory consumed in both training and testing for all above approaches

We previously created two files; one to detect anomaly based attacks and one to detect misuse based attacks. These files had about 4500 instances which we continue to use for this phase and this input is divided into a Training Data Set (75%) to train our Neural Network and Test Data Set (25%) on our trained Neural Network.

Preprocessing

The first objective in our approach was to simplify the data that is to be processed. Simplifying would mean removing attributes that did not make sense. The advantage is that removing attributes would reduce the size of data getting processed which would increase the performance of the neural network. The downside to this can be if important attributes are accidently removed then the accuracy of detecting an intrusion will suffer. The way out of this was using various iterations of removing certain attributes and then using certain attributes to figure out what suits best. We used Weka's RemoveUseless() that helps remove attributes that usually do not vary much. All constant attributes are removed and attributes that exceed the maximum percentage of a variance parameter are also removed. Our "R" scripts also checks the quality of an attribute based on variance in the attribute samples. Attribute reduction was carried out for attributes that do not contribute even 1 percent of the cumulative variation in the data set.

Methods Used

DataSet:

For carrying out intrusion detection for Anomaly based attacks and Misuse based attacks we had two data sets *Dataset_Anomaly.csv* and *Dataset_Misuse.csv* in the previous phase. In the anomaly detection data set, the class or prediction variable is either ***Normal*** which represents a normal case or an ***Attack.*** Contrary to the anomaly detection data set, the misuse detection data set has a class variable ***Normal*** or ***Name of the attack*** which represents a specific type of attack such as Smurf, NMap, Rootkit, etc. We carry out data cleaning on *Dataset_Anomaly.csv* and *Dataset_Misuse.csv* using Weka's to obtain *Dataset_Anomaly_AttributeSelection.csv* and *Dataset_Misuse_AttributeSelection.csv* which has less attributes that help speed our NN.

Anomaly and Misuse IDS operation using NN:

The 'neuralnet' package is available in R and is open source. It was used for our neural network based IDS and Analysis. The package provides functions to both generate the neural network and carry out classification. Our attribute values were used to create formula that is supplied to 'neuralnet' function. The neuralnet function returned an object that has all relevant information about the neural network and can be further used to derive our results.

Implementation

For classification we took into consideration 4500+ instances of normal cases and attack cases. We chose 10 types of attacks including Neptune, NMap, PortSweep, Satan, Smurf, BufferOverflow, FTPWrite, GuessPassword, Back and Rootkit attacks.

Input to our "R" scripts were as follows:

Dataset_Anomaly_AttributeSelection.csv to **Anomaly.R** :
This script implements Anomaly based Intrusion Detection to provide Confusion Matrix, Classification Accuracy, Time taken for implementation and Resource Consumption. It classifies if there is Attack/Normal Case

Dataset_Misuse_AttributeSelection.csv to **Misuse.R:**
This script implements Anomaly based Intrusion Detection to provide Confusion Matrix, Classification Accuracy, Time taken for implementation and Resource Consumption. It classifies between 10 Atacks/Normal Case

Dataset_Misuse_AttributeSelection.csv to **Individual.R**:
This script implementes Misuse Detection System Capable Of Detecting A Particular Attack to provide Confusion Matrix, Classification Accuracy, Time taken for implementation and Resource Consumption for 10 individual attacks. It classifies between a Particular Attack/ Other Cases

Procedure

- Open WEKA and select the Explorer Mode
- Open *Dataset_Anomaly.csv* and *Dataset_Misuse.csv* which are our datasets from Phase1
- Under **Preprocess** data tab select :
  Filter>Unsupervised>Attribute>RemoveUseless()
- We generate new datasets after saving from WEKA as *Dataset_Anomaly_AttributeSelection.csv and Dataset_Misuse_AttributeSelection.csv*
- Download R from ***http://www.r-project.org/***
- Open R and select Open Script to select new datasets
- Select all the code and press "Run" to execute script commands
- The script uses a nueralnet library which will train using 75% Training Set
- Tests are carried out on a 25% Test set to generate results such as summary, confusion matrix and accuracy and we can determine the number of false positives and false negatives
- The results outputted are as shown in the snapshots

Results

The following are results of running our R scripts on the new datasets. Summary of results is given below and detailed runtime report is submitted in a file called *DetailedRuntimeSummary*

Anomaly Detction using NN

	Axis2	
Axis1	Attack	Normal
Attack	387	4
Normal	1	759

```
IDS Accuracy: 99.57 %
Execution Time: 3.9982 seconds
Memory Usage: 2191.312 Kbs
```

Misuse Detection using NN

	Axis2										
Axis1	Back	BufferOverflow	FTPWrite	GuessPassword	Neptune	NMap	Normal	PortSweep	Rootkit	Satan	Smurf
Back	69	0	0	0	0	0	0	0	0	0	0
BufferOverflow	0	5	0	0	0	1	1	0	0	0	0
FTPWrite	0	0	1	1	0	0	0	0	0	0	0
GuessPassword	1	0	0	10	0	1	0	0	0	0	0
Neptune	0	0	0	0	54	0	0	1	0	0	2
NMap	0	0	0	0	0	72	0	0	0	0	0
Normal	0	0	0	0	0	0	744	0	1	0	0
PortSweep	0	0	0	0	0	0	0	60	0	0	1
Rootkit	0	0	1	0	0	0	1	2	0	0	1
Satan	0	0	0	0	2	2	0	1	1	58	1
Smurf	0	0	0	0	0	0	0	0	0	0	56

```
IDS Accuracy:     98.09 %
Execution Time:   48.9288 seconds
Memory Usage:     2988.16 Kbs
```

Misuse Detection for Individual Attacks (10 attacks):

```
Attack :   Neptune
           Axis2
Axis1        Neptune OtherCase
   Neptune         57         0
  OtherCase         0      1094
IDS Accuracy:    100 %
Execution Time:  7.1994 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   Satan
           Axis2
Axis1        Satan OtherCase
   Satan        12         48
  OtherCase      8      1083
IDS Accuracy:    95.13 %
Execution Time:  1.3051 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   Smurf
           Axis2
Axis1        Smurf OtherCase
   Smurf        65          1
  OtherCase      0       1085
IDS Accuracy:    99.91 %
Execution Time:  0.459 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   PortSweep
           Axis2
Axis1          PortSweep OtherCase
   PortSweep           3         58
  OtherCase            2       1088
IDS Accuracy:    94.79 %
Execution Time:  1.0751 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   NMap
           Axis2
Axis1        NMap OtherCase
   NMap         56          1
  OtherCase      3       1091
IDS Accuracy:    99.65 %
Execution Time:  5.2433 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   BufferOverflow
                  Axis2
Axis1               BufferOverflow OtherCase
   BufferOverflow                6         0
  OtherCase                      1      1144
IDS Accuracy:    99.91 %
Execution Time:  0.412 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   FTPWrite
           Axis2
Axis1       OtherCase
  OtherCase       1151
IDS Accuracy:    100 %
Execution Time:  0.313 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   GuessPassword
                Axis2
Axis1             GuessPassword OtherCase
   GuessPassword             11         0
  OtherCase                   1      1139
IDS Accuracy:    99.91 %
Execution Time:  2.5021 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   Back
           Axis2
Axis1        Back OtherCase
   Back        61         1
  OtherCase     1      1088
IDS Accuracy:    99.83 %
Execution Time:  0.173 seconds
Memory Usage:    2191.312 Kbs
```

```
Attack :   Rootkit
           Axis2
Axis1       OtherCase
   Rootkit          2
  OtherCase       1149
IDS Accuracy:    0.17 %
Execution Time:  0.183 seconds
Memory Usage:    2191.312 Kbs
```

Accuracy Summary:

Classification Approach	Accuracy for Anomaly based detection	Accuracy for Misuse based detection
Using nueralnet package in R	**99.57**	**98.09**

Result Summary:

The results that we have above are drawn from running tests for 3 approaches, the summary and snapshots are attached above. The original dataset that we had with 41 attributes which were reduced to 36 using Weka's RemoveUseless() and then eliminating other redundant attributes with low variance. If we ignore the difference in the number of these attributes everything else in the dataset is just same which is why we have similar scripts for these different data sets.

We have 10 attack types, the output of each is in the form of a confusion matrix. The matrix helps us analyze false positives and negatives since it has the following representation:

True Positives	False Negatives
False Positives	True Negatives

The output also talks about the other features like the consumed amount of memory in Kbs, the time taken in seconds and the accuracy of classification in percent.

Development Process and Work Distribution

We tried our best to divide the work equally and work to each other's strengths. The datasets used in the implementation are build using a Java Program. Weka is used to remove useless attributes and then R is used for the neural network implementation, using the 'neuralnet' package. The datasets created were with specifics in mind about the implementation. So there 3 major specifications for which each member created a specific dataset. The distributions of work can be specifically listed out as:

Member 1	- Using weka to remove useless attributes - Preparing dataset for Anomaly detection
Member 2	- Writing a generic Java program to remove attributes - Preparing the dataset for Misuse detection.
Member 3	- Using R for neural network implementation - Dataset for distinguishing between attack and not an attack
Group Work	- We met once a week initially to discuss our progress and then it increased gradually. - Discussion about removal of unwanted attributes - Deciding which method to use for neural network implementation. - Creating the final project report.