NETWORK INTRUSION DETECTION SYSTEM

WEB SECURITY (BCI3001)
SLOT L41+L42
PROJECT COMPONENT - REVIEW REPORT

Submitted By

Vagisha Srivastava (16BCS0079)

Submitted To

Prof. D Anuradha
Assistant Professor SCOPE

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

November, 2018

TABLE OF CONTENTS

1. ABSTRACT

This project deals with practical implementation of the theoretical solutions to the detection of network intrusion. Techniques tested for the final implementation included graph reduction, fuzzy logic and other algorithms before finally settling in for neural network algorithm for correct implementation. The accuracy percentage for neural network algorithm was approximately 99% in case of anomaly based detection and 98% for misuse based detection.

2. INTRODUCTION

In this project, I have implemented Intrusion Detection System using Neural Network. There are various phases in which this project was divided for better implementation and smoother result. Part of the data was manually collected and other was taken from kaggle's database on network intrusion (The link is available in reference section.) Tools like weka was used for the preprocessing and data cleaning in phase two. The output obtained from phase two was then used for the final phase in which neural net package (available in R) was used to predict and classify the anomaly and misuse. The dataset obtained from Phase two was categorized into training and testing sets to train the Neural Network algorithm and perform the test to detect the intrusion.

3. METHODOLOGY

The complete project was divided into three basic phases. The first phase included literature review on the matter. Various techniques were explored in this phase to figure out which way this project could turn. The second phase included data collection, cleaning and pre-processing. The data was collected from multiple sources but ultimately from kaggle's database set.

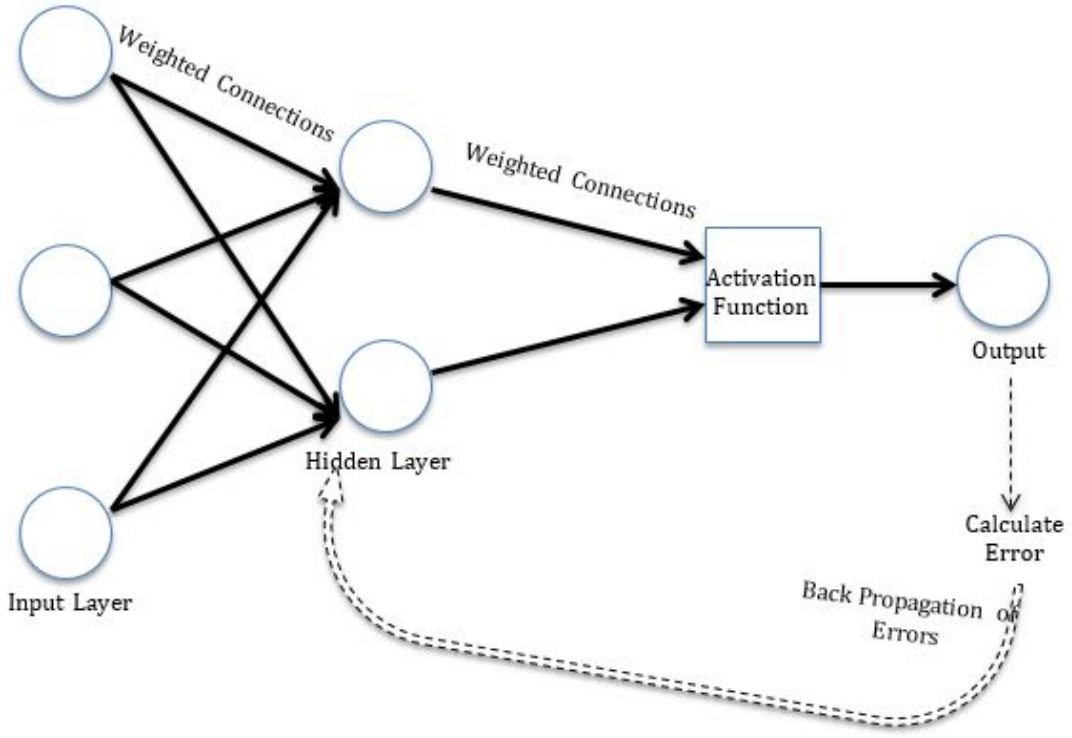
The third and the last phase included using neural network algorithm on the output data from the second phase. The dataset was divided into two - training and testing sets which were worked upon respectively.

Ten types of attacks are included in this dataset, namely, **NMap, PortSweep, Satan, Smurf, BufferOverflow, FTPWrite, GuessPassword, Back** and **Rootkit** attacks**.**

Preprocessing - Simplification of the data to be processed is very important. In data cleaning, we remove the attributes that are otherwise useless. This makes an advantage of reducing the size and hence increasing the speed of neural network. The negative of this, although less, is also there. If by mistake we remove an important attribute, that would affect the accuracy of the model. Multiple test approach to this gave a correct implementation solution. The tools used for preprocessing is **WEKA**. The RemoveUseless() function is very helpful for this. It removes the constant attributes along with the attributes with maximum variance.

Before running the above data through neural network, **R** is used to check the quality of the dataset (based on variance.)

Neural Network in R : Neural Network is a model characterized by an activation function, which is used by interconnected information processing units to transform input to output. [1]
The first layer of the network received the raw input, processes it and pasees the processed informaton to the hidden layers. The hidden layers pass the information to the last layer, which produces the output.



2 A simple neural network model.

4. IMPLEMENTATION

Requirements - There are no hardware requirements to run this project. The software use and details are mentioned below.
Key List

1. Weka Tools [v. 3.9.3] [3]
2. Java [v.8]
3. R [v. 3.5.1] [4]
4. Libraries in R
    a. kernlab

[1] Text taken from : Analytics Vidhya
[2] Image credit : Analytics Vidhya [https://www.analyticsvidhya.com/blog/2017/09/creating-visualizing-neural-network-in-r/]
[3] Weka can be downloaded from - https://www.cs.waikato.ac.nz/ml/weka/
[4] R can be downloaded from - http://www.r-project.org/

b. caret
c. neuralnet

There is no specific system requirement. However, it's important to mention here that I use a linux system (Ubuntu 18.04) and things worked well on that. Transitioning it to some other system might cause some issues.

Sample Code :

Java snippet for file arrangement

```
// Reads inputs and creates a list that represents data in the file
      public static void generateFile(String input, String output,
String column,
                  int size) throws FileNotFoundException, IOException {
            BufferedReader br = new BufferedReader(new
FileReader(input));
            String line;
            int i = 0;
            while ((line = br.readLine()) != null && i < size) {
                  list.add(line + column);
                  i++;
            }
            br.close();
      }

      // swaps array elements i and j
      public static void swap(String[] a, int i, int j) {
            String swap = a[i];
            a[i] = a[j];
            a[j] = swap;
      }

      // takes as input an array of strings and rearranges them in
random order
      public static void printShuffle(String[] a, String output)
                  throws FileNotFoundException,
UnsupportedEncodingException,
                  IOException {
            int N = a.length;
            for (int i = 0; i < N; i++) {
                  int r = i + (int) (Math.random() * (N - i)); //
between i and N-1
                  swap(a, i, r);
            }
            File file = new File(output);
```

```
            Writer writer = null;
            if (!file.exists()) {
                  writer = new BufferedWriter(new OutputStreamWriter(
                              new FileOutputStream(output), "utf-8"));
            } else
                  writer = new PrintWriter(new BufferedWriter(new
FileWriter(output,
                              true)));
            // Attributes list
            writer.write("duration, protocol_type, service, flag,
src_bytes, dst_bytes, land, wrong_fragment, urgent, hot,
num_failed_logins, logged_in, num_compromised, root_shell, su_attempted,
num_root, num_file_creations, num_shells, num_access_files,
num_outbound_cmds, is_host_login, is_guest_login, count, srv_count,
serror_rate, srv_error_rate, rerror_rate, srv_rerror_rate,
same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count,
dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate,
dst_host_same_src_port_rate, dst_host_srv_diff_host_rate,
dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate,
dst_host_srv_rerror_rate, AttackType\n");
            for (int i = 0; i < N; i++) {
                  writer.write(a[i]);
            }
            writer.close();
}
```

R code for Misuse and Anomaly

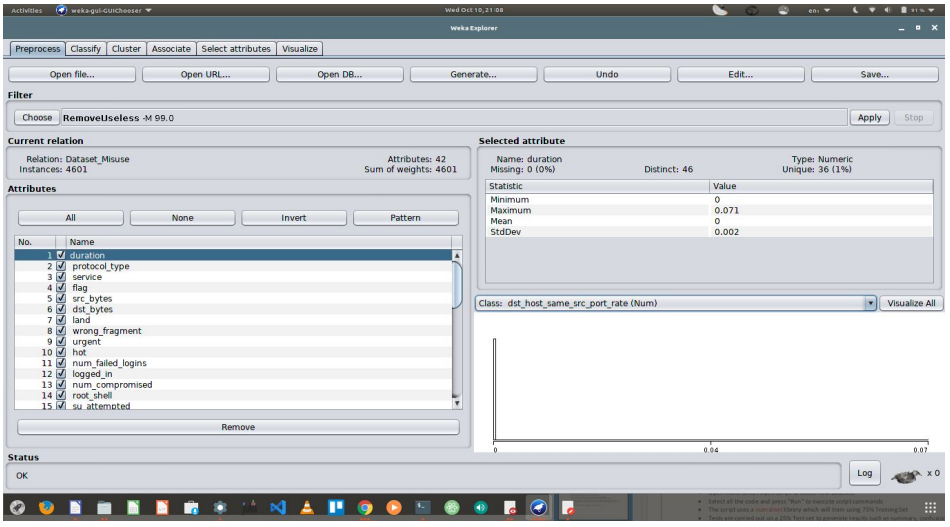
This code is used to generate confusion matrix to predict and classify data.

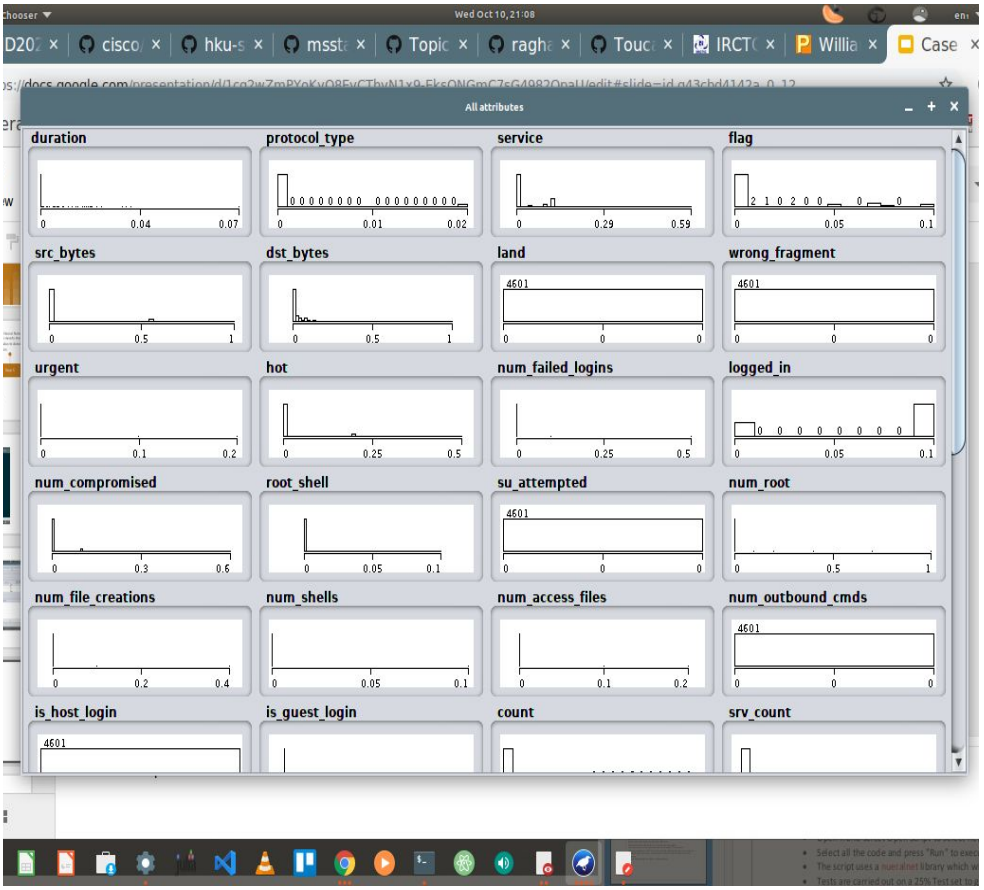
```
library(kernlab)
library(caret)
anomaly<-read.csv("/home/vagisha/Desktop/Classes/College/Semester 5/Web
Security/Project
/Network-Intrusion-Detection-System/data/Dataset_Anomaly.csv",
na.strings=c(".", "NA", "", "?"), strip.white=TRUE, encoding="UTF-8")
aRow<-nrow(anomaly)
aCol<-ncol(anomaly)

sub<-sample(1:aRow,floor(0.66*aRow))
anomalyTrainingSet<- anomaly[sub,]
anomalyTestSet<- anomaly[-sub,]
anomalyClassifier<- ksvm(AttackType~.,data=anomalyTrainingSet,type =
'C-svc', kernel = 'rbfdot')
anomalyPrediction<-predict(anomalyClassifier, anomalyTestSet[,-aCol])
confusionMatrix(anomalyPrediction,anomalyTestSet[,aCol] )
```

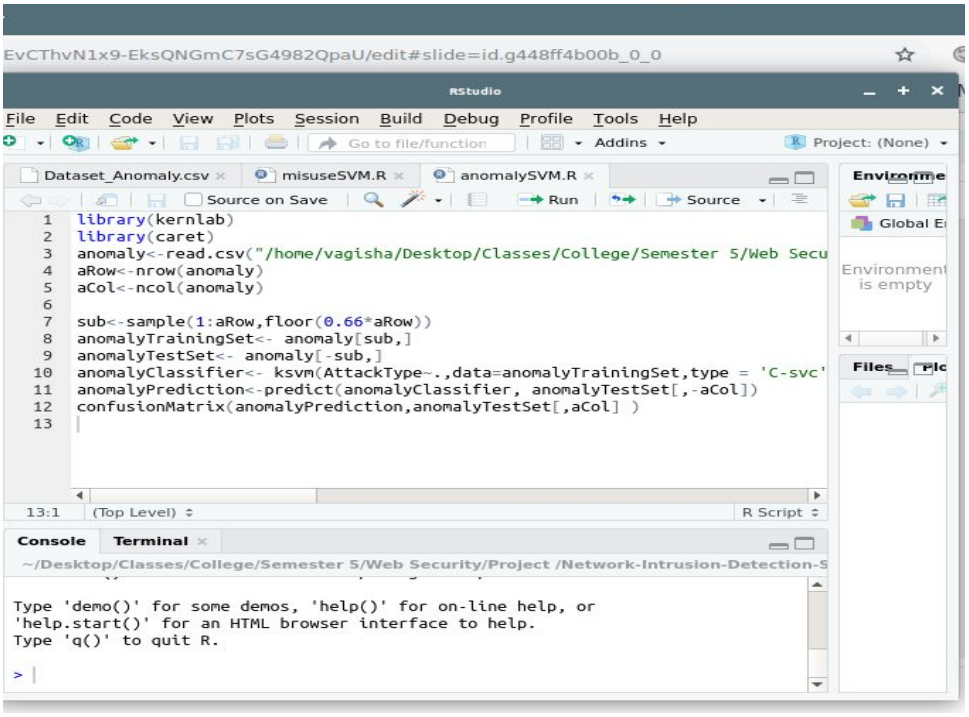
Snapshots

Weka





R Studio



5. VULNERABILITY ANALYSIS

A dataset with 4600+ instances were used. This case set included both normal and attack cases. The types of attacks includes - NMap, Rootkit, Neptune, Smurf, FTPWrite, GuessPassword, Satan, Back, PortSweep, BufferOverflow and Rootkit.

The R script (code above) implements Anomaly Based Intrusion detection and Misuse to provide Confusion matrix, time taken, classification accuracy and resource consumption. The first one is used to classify a case of Attack/normal case while the latter is used to classify it out of the 10 attacks mentioned.

6. RESULT

Running the codes mentioned above gives the following output. The summary is presented below. For detailed result view, kindly run the code provided with documentation on my github handle.[5]

[5] Github Handle - Vagisha@16

Anomaly Detection using neural network

	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 Neural Network

	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
```

For individual attack

```
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 :   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 :    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 :    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
```

```
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
```

Accuracy

Classification	**Anomaly Based**	**Misuse based**
Using Neural Network	99.57	98.06

7. PREVENTIVE MEASURES

Intrusion prevention is considered by some system users as an extension of the intrusion detection technology. This term was initially coined by the Andrew Plato who was technical consultant and writer for Network ICE. However, there are various ways in which one can prevent intrusion in the system.

The very first step in intrusion prevention is front door also known as firewall. This should be locked to avoid strangers from helping themselves with ones virtual treasures such as bank accounts, password, personal information and credit cards.

This type of intrusion prevention has two ways of avoiding intruders from visiting ones front door. These are prevention software and prevention hardware.

The second step is by software designed intrusion prevention to reinforce one's web security. The software appears in two basics, flavors and improved web security audits. The web security services are ideal for average PC users, easy to use and also to understand. These services for web security are concentrated audits that are designed with network administrator in them. They also have more options and they also include ability to make audit to one's particular need. The user should make sure that the software is operating correctly by testing free firewall. This helps the user to find out the program that can work with the ports.

The third step used in Intrusion prevention can be done by using hardware, in this case, Cisco or NetGear help ease the intrusion prevention by configuring direct tight web security. These hardware devices are not wireless but land based. One should note that wireless web devices are not well configured out of the box like land based devices.

The fourth step used in intrusion prevention is by use of ADS. Though the front door is locked some malicious website can still try to trick one in acquiring special treatment by using tricky pop ups that say "how to turn off security services on the web" and give an option of YES or NO. By selecting the option, the intruder gets access to one computer.

The final step is by installing spyware removal programs and adding a virus prevention program to back it up. By using the above given steps, one is able to adequately prevent intrusion in the computer and not worry about anyone else gaining access to the computer. [6]

8. CONCLUSION

To conclude this report, all the source for the survey is provided in the reference section and additional details are mentioned in the footnotes. The project was successfully completed with proper implementation. All the code is uploaded on my Github handle for any implementation purpose. The inspiration for this project was another similar project implemented on python. Help was found from papers published on the subject.

The output for each attack type was in the form of confusion matrix. The matrix was helpful to categories the result in false positive, false negative, true positive and true negative.

[6] Source : http://www.liutilities.com/articles/5-easy-steps-intrusion-prevention/#.W-pVm8szbNw

References

http://www.csm-ace.my/presentation/Day1_Track2_2012/Neil%20Meikle_%20Case%20Study%20Big%20Data%20Forensics.pdf

http://www.studioag.pro/en/2013/10/big-data-e-digital-forensics/

https://articles.forensicfocus.com/2017/08/07/digital-forensics-as-a-big-data-challenge/

https://pdfs.semanticscholar.org/aa9e/79133f619bfa502d7fc7470446d7a8b5d0cf.pdf

https://www.researchgate.net/publication/281257770_Digital_Forensics_in_the_Age_of_Big_Data_Challenges_Approaches_and_Opportunities

https://ieeexplore.ieee.org/document/7351932/

https://www.analyticsvidhya.com/blog/2017/09/creating-visualizing-neural-network-in-r/

http://www.liutilities.com/articles/5-easy-steps-intrusion-prevention/#.W-pVm8szbNw