CSCI - 735 Foundations of Intelligent Security Systems

Project 1

Intrusion Detection System Design and Evaluation

Submitted By

Dhivya Govindarajan (dng2378@rit.edu)
Jaydeep Untwal (jmu8722@rit.edu)
Bhakti Bhadrecha (bjb4484@rit.edu)

Executive Summary

Technology is evolving everyday, we have become heavily dependent on it for almost all our activities that were once time consuming and involved physical presence, e.g, depositing checks in bank accounts, signing contracts, etc. but with this convenience, comes great responsibility. This has made us vulnerable to malicious and tech savvy criminals to get unauthorized access to our confidential information. Just as we had security guards protecting our apartments and neighborhood banks, we need virtual guards to protect our information online.

This project aims to design and implement an intrusion detection system (the above mentioned virtual guard) to monitor and alert us when it notices any suspicious activities like unauthorized access, modification of protected files and user spoofing. This will be achieved by analyzing network data of known attacks and normal behavior, which will help us prevent the system from future attacks. WEKA is an easy to use, open source application that allows us to perform several data analytics tasks. We will be embedding the WEKA library in a Java program of our Intrusion Detection system**.**

Specification

This section of the report describes about the specifications of the Intrusion Detection System. The team analyzed and researched about several tools such as Snort, WEKA etc for Intrusion Detection System and decided WEKA as the tool for this project. The reason for eliminating Snort is because it doesn't offer several data mining techniques and better data visualization that WEKA does. WEKA is a repository of machine learning algorithms designed for the purpose of data mining. It is an open-source software and written in JAVA. It contains various algorithms for data preprocessing, classification, clustering, regression and data visualization. To build a successful detection model, it is significant to understand how the model is trained and validated. The tool splits the data into two sets: training and test data. The training dataset is a n-dimensional vector of attributes which is used to build and train the model of potential predictions. The test dataset is used to assess and validate the model built. Most of the data is used as the training data and the rest of the data portion are used for testing. WEKA Explorer offers this feature of splitting the data. In a nutshell, the few of the many compelling features that made us to choose WEKA for this project are as follows: the feature that allows us to analyze, preprocess and validate the correctness of the data, the wide range of algorithms that it offers for data visualization and data mining, the feature that allows us to select the data attributes for the mining, the better visualization of the results through graphs, trees, plots, clusters, etc., the easy-to-use graphical user interface and many more features.

To classify and analyze the pattern of the dataset, we decided to use the J48 classifier. J48 is the implementation of the C4.5 algorithm in the WEKA. It is open source and written in JAVA. This algorithm is a predictive machine learning model which generates the decision tree of a given dataset based on the information gain value of an attribute or feature. The information gain is the measure of the significance of the attributes in a given dataset i.e. it helps in determining which attribute is the most useful for the classification. The results are in a tree-shaped structure. Each internal nodes of the tree represents the attributes or features of the data, the branches represents the values of the attributes and the terminal nodes represents the classification results. The algorithm works by finding the information gain value for each attributes. Then, it chooses the attribute with the highest normalized information gain. The decision tree node then splits on the attribute chosen based on the information gain. Now, it recursively calculates the information gain for the rest of the attributes and add the next best attribute as the child nodes. The results are visualized in the form of a tree which helps us get better insight of the results.

The main objective of this project is to design a Intrusion Detection System that identifies and detects the misuse and anomaly based attacks. This could be achieved by collecting the real time network data, preprocessing the collected data for analysis, then analyzing the behavior or the pattern of the data and building a detection model using the classification algorithms, that would notify the system whenever an unusual or abnormal behavior is noticed. To validate the model, we are analyzing the false positives and false negatives. These are two significant metrics that are used to evaluate the performance of the model. They are the measure of incorrectly classified instances. The false positives and false negatives are discussed in the Results section.

Methods and Techniques

This section explains the methods and techniques employed in this project. The data files provided in the Project section of the mycourses are used for this project. It comprises of 23 data files out of which 22 are different kinds of attacks such as Back, Buffer Overflow, FTPWrite, GuessPassword, Imap, IPSweep, Land, LoadModule, MultiHop, Neptune, NMap, Perl, PHF,Pod, PortSweep, RootKit, Satan, Smurf, Spy, TearDrop, WarezClient and WarezMaster. Each file consists of 41 attributes and several data records. There are nine basic attributes: Duration, Protocol Type, Service Type, Status Flag, Total bytes sent to Destination & Source, etc. and 32 derived attributes which are divided into three categories: Content, Time-based Traffic and Host-based Traffic. In this project, we considered to build a detection/classification model that detects/classifies all of the attacks mentioned above. The figure 1 shows the count of the records in each file.

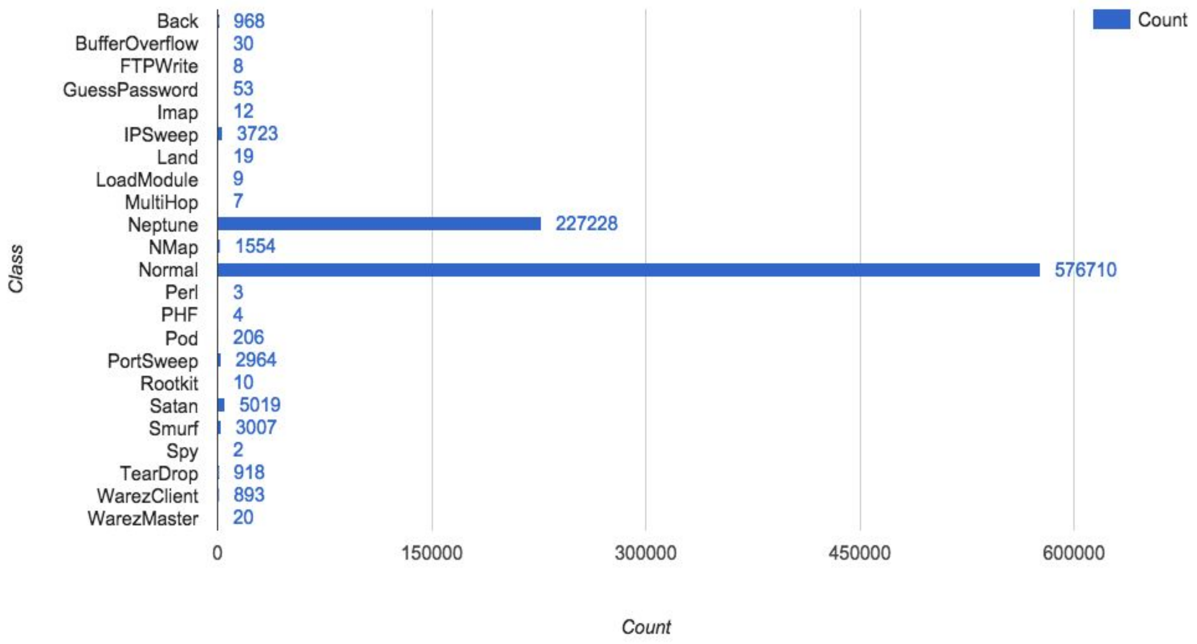


Figure 1 Count of the data records

The first step involved in the data preprocessing was to clean the data. The basic data cleansing was performed, the unique ids and '0' valued attributes, because they did not influence the results, were removed from the dataset. The next step was to generate the Misused datafile and the Anomaly datafile. Most of the data preparation were done using the R tool. The R script was written to perform the following operations: to add the target/class attributes, to extract the data records, to minimize the data records from the Neptune and Normal data file as those files had the highest number of records (as shown in figure 1), to randomize the order of the extracted data records from each file and finally to generate the Misused and Anomaly datafiles. The generated Misused and Anomaly files were checked and validated once again and then fed to WEKA for the classification. The Misused datafile has the target/class attributes whose values ranges between different kinds of attacks (E.g. Class = Buffer Overflow, etc.). and the normal behavior type (E.g. Class = Normal). Similarly, the Anomaly datafile consists of a target/class attributes where the values would be attack, if the data record has the attack behavior and normal, if the data record has the normal behavior. The files were in the .csv format which works well with WEKA. The Misused and the Anomaly data files were split into 70% of training data to train the model and 30% of test data to test the built model. The files were then fed to WEKA to build the classification models. As a result, we obtained the Misuse-Based classifier model and the Anomaly-Based classifier model. The results of the classification model,in our case J48 classifier, are explained in the Results section. The figure 2 provides a simple overview of the project process flow.

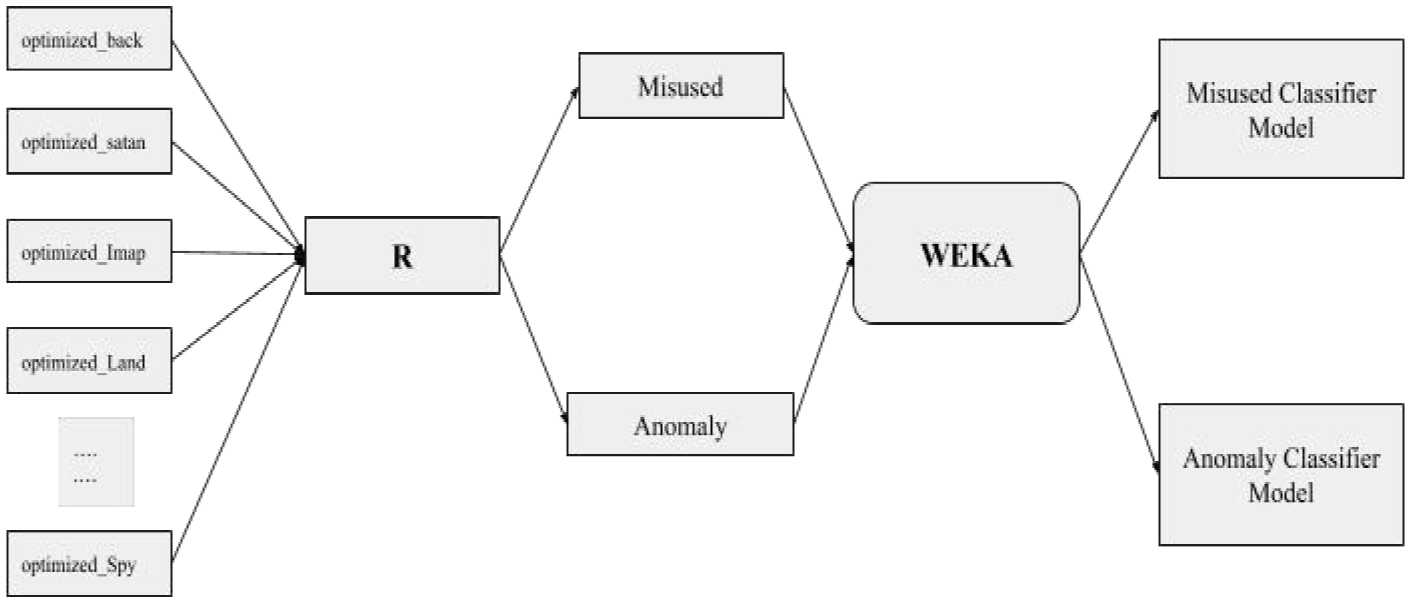


Figure 2 Project Process Flow Diagram

Implementation

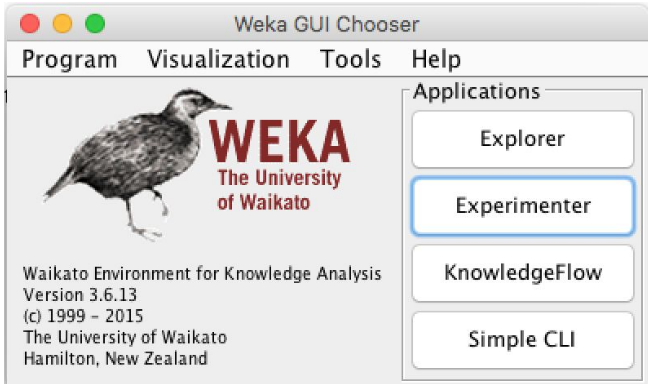
We had a dataset of TCP dumps classified into attacks and normal behavior.
Using the 'R' tool, we wrote a script to extract records from the original dataset and generate two new datasets:

1. anomaly_shuffle.csv
2. misused_shuffle.csv

In order to generate the classifier model and the decision tree, we used the WEKA classification tool.

Following are the steps performed to generate the decision tree:

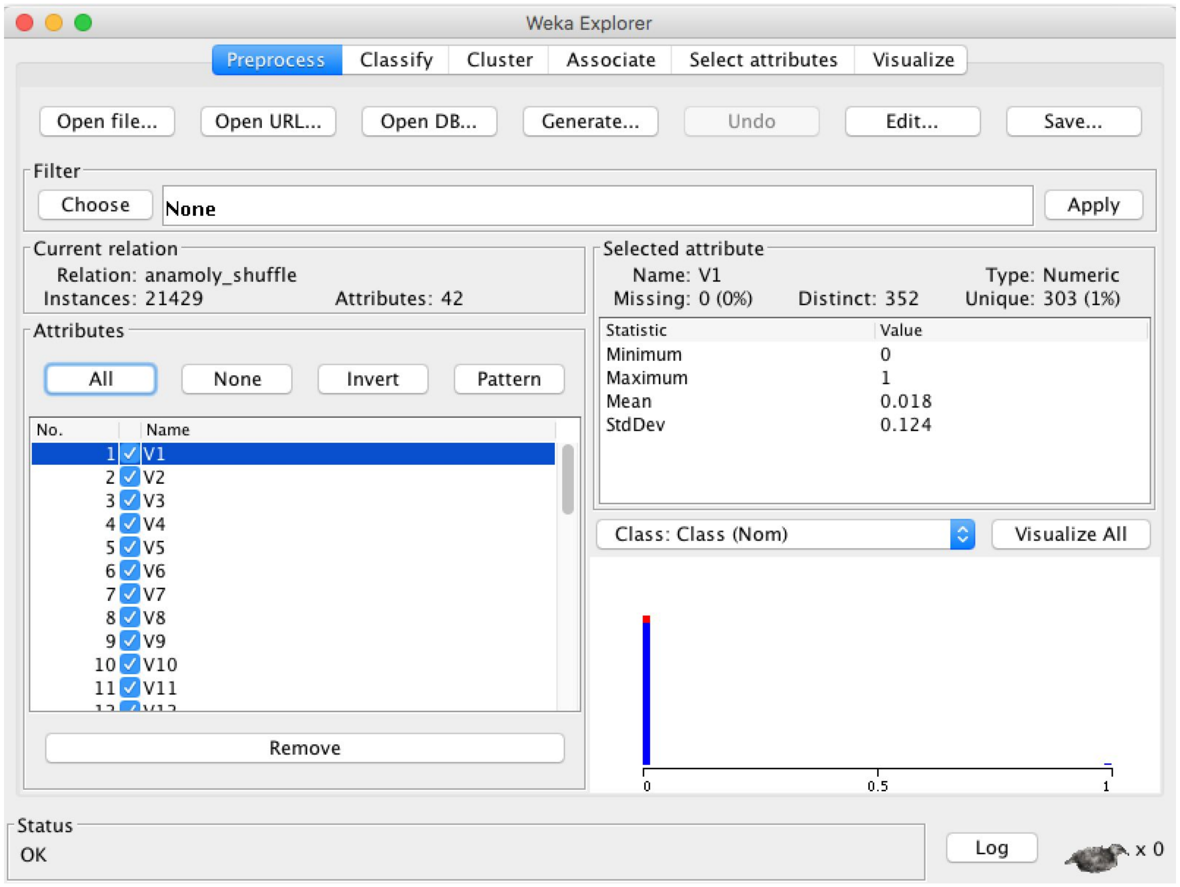
1. Open WEKA and click on 'Explorer'



Waikato Environment for Knowledge Analysis
Version 3.6.13
(c) 1999 - 2015
The University of Waikato
Hamilton, New Zealand

Figure 3 WEKA GUI Chooser

2.
   a. Click on 'Open File', select your dataset and click 'Ok'
   b. Once the dataset is loaded, you can click on the 'Classify' tab



Filter: None

Relation: anamoly_shuffle
Instances: 21429 Attributes: 42

No.	Name
1	✓ V1
2	✓ V2
3	✓ V3
4	✓ V4
5	✓ V5
6	✓ V6
7	✓ V7
8	✓ V8
9	✓ V9
10	✓ V10
11	✓ V11

Name: V1 Type: Numeric
Missing: 0 (0%) Distinct: 352 Unique: 303 (1%)

Statistic	Value
Minimum	0
Maximum	1
Mean	0.018
StdDev	0.124

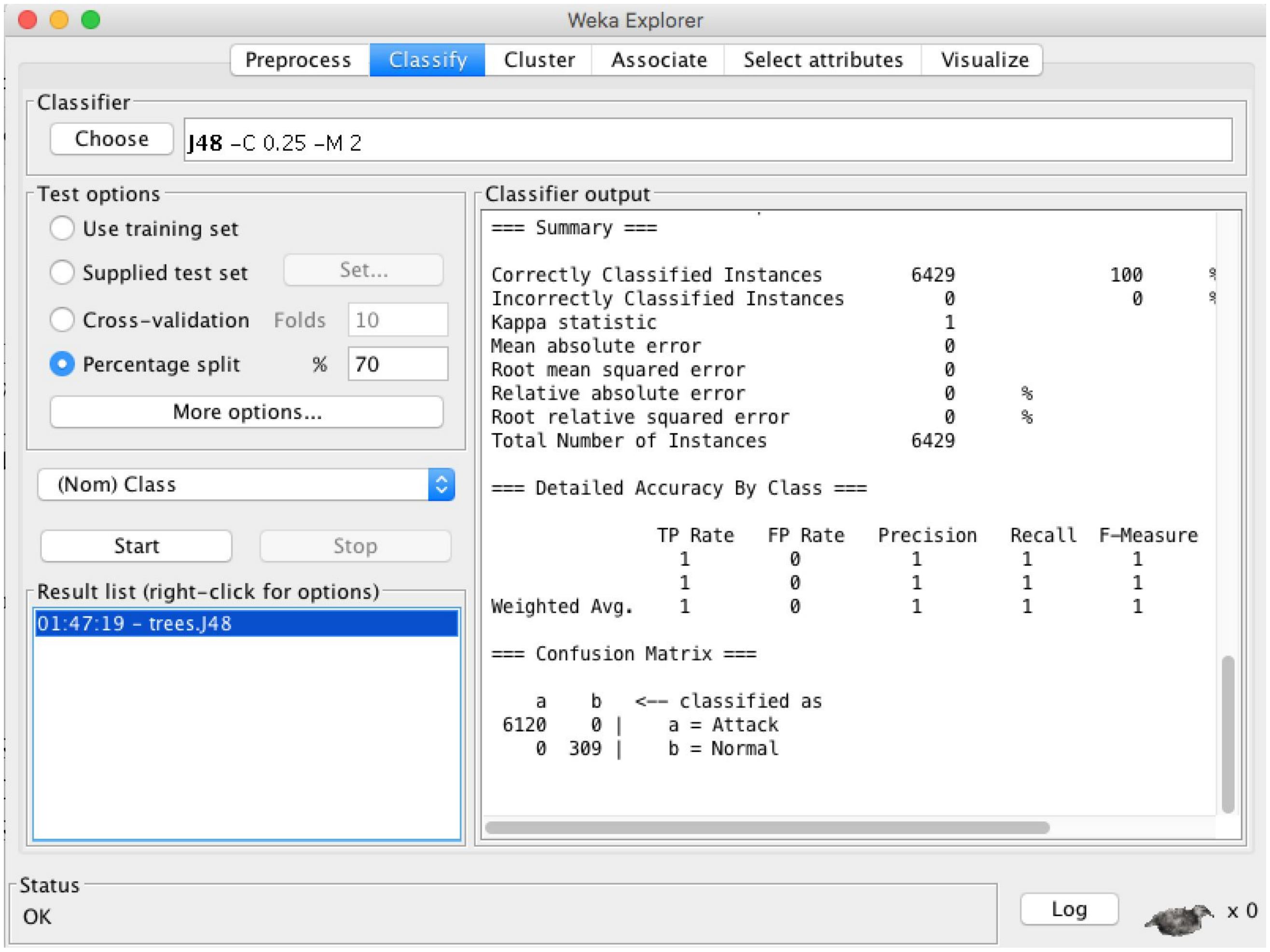
Class: Class (Nom)

Status: OK

Figure 4 Loading dataset in WEKA

3.

   a. Click on ‘Choose’ and select ‘Trees > J48’
   b. Select Percentage split and enter value ‘70’
   c. Choose a class in the drop down and select your class attribute
   d. Click on ‘Start’ and wait for results
   e. Once the results populate, you can right click on the generated model and click on ‘Visualize tree’



J48 -C 0.25 -M 2

Cross-validation Folds 10

Percentage split % 70

(Nom) Class

01:47:19 - trees.J48

```
=== Summary ===

Correctly Classified Instances        6429               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances             6429

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure
                 1         0          1          1         1
                 1         0          1          1         1
Weighted Avg.    1         0          1          1         1

=== Confusion Matrix ===

    a    b   <-- classified as
 6120    0 |    a = Attack
    0  309 |    b = Normal
```

Status
OK

Figure 5 Running J48 classifier in WEKA

Results

- **Anomaly-Based:**

The results shown below were obtained by splitting the anomaly-shuffle.csv into 70% training and 30% test data. The classifier model resulted in 100% accuracy in classifying 6429 instances. Hence, all of the attack instances are classified as “attack” and all of the normal instances are classified as “normal”. ‘V3’ was chosen as the best attribute to initially split on, as it is shown in the decision tree. The weighted false positive rate was ‘0’ and the weighted true positive rate was ‘1’

```
=== Classifier model (full training set) ===
            J48 pruned tree
            ------------------

            V3 <= 0
            |   V5 <= 0.06274
            |   |   V5 <= 0.0009: Attack (34.0)
            |   |   V5 > 0.0009: Normal (1000.0)
            |   V5 > 0.06274: Attack (968.0)
            V3 > 0: Attack (19427.0)

            Number of Leaves  :     4
            Size of the tree :     7

            Time taken to build model: 0.27 seconds

            === Detailed Accuracy By Class ===

                       TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                         1         0          1          1         1          1       Attack
                         1         0          1          1         1          1       Normal
            Weighted Avg.   1         0          1          1         1          1

            === Evaluation on test split ===
            === Summary ===
            Correctly Classified Instances        6429              100      %
            Incorrectly Classified Instances         0                0      %
            Kappa statistic                          1
            Mean absolute error                      0
```

```
Root mean squared error              0
Relative absolute error              0  %
Root relative squared error          0   %
Total Number of Instances            6429

=== Confusion Matrix ===

   a    b   <-- classified as
 6120    0 |    a = Attack
   0  309 |    b = Normal
```

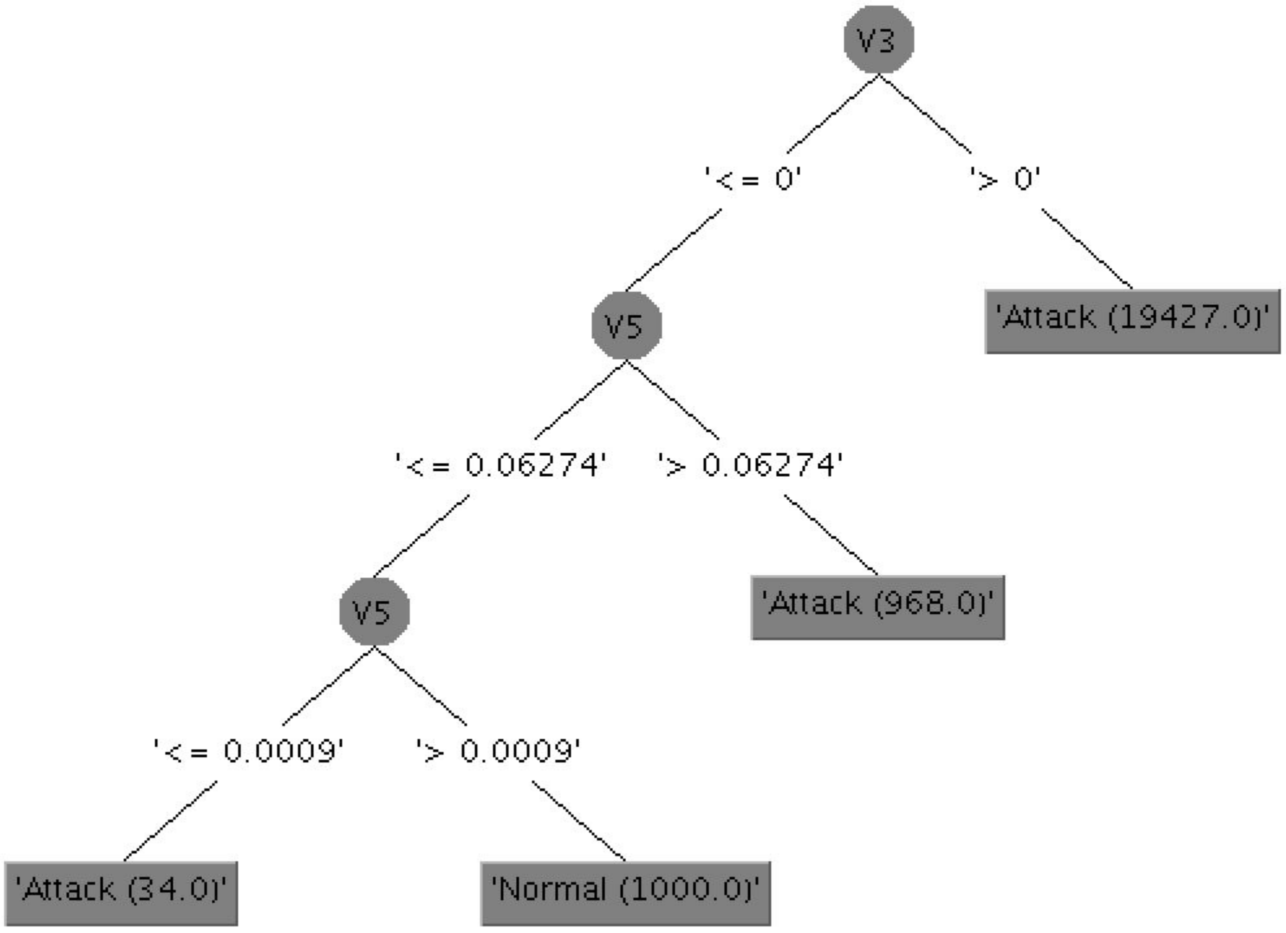


Figure 6 The anomaly-based IDS decision tree

- **Misuse-Based:**

The results shown below were obtained by splitting the misuse-shuffle.csv into 70% training and 30% test data. The classifier model resulted in 99.4711 % accuracy in classifying 6429 instances with very minimal misclassified instances. 'V12' was chosen as the best attribute to initially split on, as shown in the decision tree. The weighted false positive rate was '0.001' and the weighted true positive rate was '0.995'.

```
Number of Leaves  :        77
Size of the tree :         153

Time taken to build model: 1.26 seconds
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances        6395              99.4711 %
Incorrectly Classified Instances        34               0.5289 %
Kappa statistic                       0.9939
Mean absolute error                    0.0006
Root mean squared error               0.02
Relative absolute error                0.823  %
Root relative squared error           10.3373 %
Total Number of Instances              6429
```

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.996	0	0.999	0.996	0.998	0.998	Satan
1	0	1	1	1	1	Pod
0.997	0.002	0.988	0.997	0.993	0.998	IPSweep
1	0	1	1	1	1	Smurf
0.999	0.001	0.997	0.999	0.998	1	PortSweep
0.833	0.001	0.667	0.833	0.741	0.916	WarezMaster
0.985	0	0.998	0.985	0.991	1	NMap
1	0	0.997	1	0.998	1	Neptune
0.992	0	1	0.992	0.996	0.996	WarezClient
1	0	1	1	1	1	TearDrop
1	0	1	1	1	1	Back
1	0	0.994	1	0.997	1	Normal
0.905	0	1	0.905	0.95	0.952	GuessPassword
0	0	0	0	0	0.499	MultiHop
1	0.001	0.667	1	0.8	1	BufferOverflow
0	0	0	0	0	0.738	LoadModule

```
                 1        0        1           1        1        1              Land
                 0.667    0        0.667       0.667    0.667    0.999          Imap
                 0        0        0           0        0        0.499          Rootkit
                 0        0        0           0        0        0.5            Perl
                 0        0        0           0        0        ?              PHF
                 0        0        0           0        0        ?              FTPWrite
                 0        0        0           0        0        0.5            Spy
Weighted Avg.    0.995    0.001    0.994       0.995    0.994    0.998
```

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n    o    p    q    r    s    t    u    v    w   <-- classified as
 1468    0    1    0    1    3    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0 |   a = Satan
    0   57    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   b = Pod
    0    0 1072    0    1    0    1    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0 |   c = IPSweep
    0    0    0  917    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   d = Smurf
    1    0    0    0  918    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   e = PortSweep
    0    0    0    0    0   10    0    0    0    0    0    0    0    1    0    0    0    1    0    0    0    0    0 |   f = WarezMaster
    0    0    7    0    0    0  454    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   g = NMap
    0    0    0    0    0    0    0  323    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   h = Neptune
    0    0    2    0    0    0    0    0  257    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   i = WarezClient
    0    0    0    0    0    0    0    0    0  273    0    0    0    0    0    0    0    0    0    0    0    0    0 |   j = TearDrop
    0    0    0    0    0    0    0    0    0    0  297    0    0    0    0    0    0    0    0    0    0    0    0 |   k = Back
    0    0    0    0    0    0    0    0    0    0    0  313    0    0    0    0    0    0    0    0    0    0    0 |   l = Normal
    0    0    2    0    0    0    0    0    0    0    0    0   19    0    0    0    0    0    0    0    0    0    0 |   m = GuessPassword
    0    0    0    0    0    2    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0 |   n = MultiHop
    0    0    0    0    0    0    0    0    0    0    0    0    0    8    0    0    0    0    0    0    0    0 |   o = BufferOverflow
    0    0    0    0    0    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0    0    0 |   p = LoadModule
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    7    0    0    0    0    0    0 |   q = Land
    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0    2    0    0    0    0    0 |   r = Imap
    0    0    1    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0    1    0 |   s = Rootkit
    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   t = Perl
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   u = PHF
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |   v = FTPWrite
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0 |   w = Spy
```

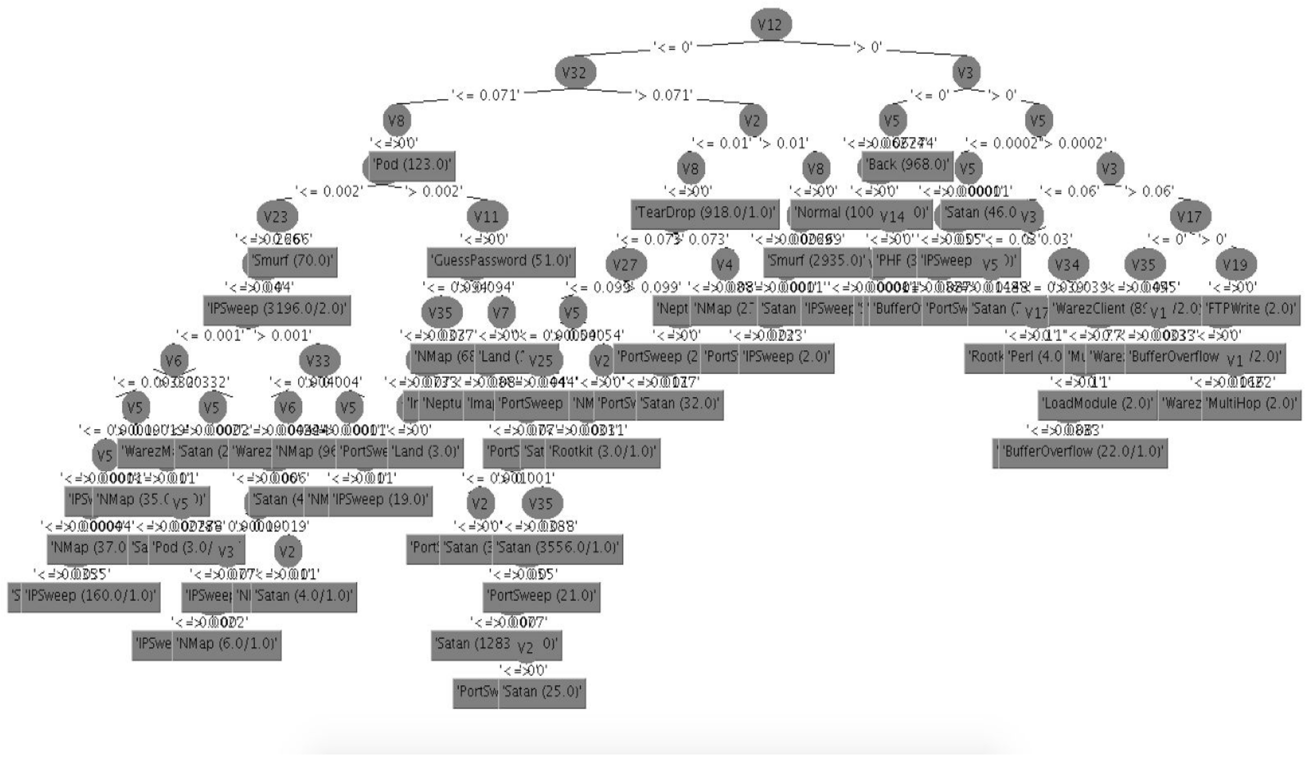


Figure 7 The misuse-based IDS decision tree

Development Process

We divided our project 1 in the following tasks:

- Research: This involved selecting the tool, understanding the dataset and designing the intrusion detection system. This was performed by all the members of the team
- Data Cleaning: Bhakti Bhadrecha performed general cleansing and removal of unique ids and '0' valued attributes from the dataset
- Data Preparation: Dhivya Govindarajan wrote a script using 'R' tool to extract the data from the original dataset and generate the anomaly and misuse dataset
- Generating Classifier / Decision Tree: Jaydeep Untwal used WEKA to generate the decision trees and classifier model using the prepared datasets and extracted the results
- Analyzing results: This involved analyzing the decision tree and results obtained by the WEKA classifier. This was performed by all the members of the team
- Documentation: This involved documenting a report explaining the above steps. This was performed by all the members of the team

References

http://www.cs.waikato.ac.nz/ml/weka/ Last accessed: 10/14/2014 12.24 PM