Network Intrusion Detection

J Geetha Reddy, Akash Gupta, Ayush Kumar, Bineet Kumar, Deepak Jayaprakash
Department of Computer Science and Engineering
M S Ramaiah Institute of Technology
Bangalore -560054, India

Abstract

With the exponential growth of internet communication technology and the number of applications running on top of it, the network traffic is increasing in accordance with it. Hence the risk posed to network security becomes a hot topic involved in research. The security and effectiveness of a wired and wireless network system are compromised due to intrusion. If the invasions are not detected at the appropriate level, the loss to system may be some times unimaginable.
Therefore, the role of Intrusion Detection Systems (IDSs), as special-purpose devices to detect anomalies and attacks in the network, is becoming more important. IDSs are impressive since it can detect, prevent and possibly react to attacks in an efficient manner.
In this paper, we present a detailed analysis of the accuracy with respect to the selection of attributes, model built and the size of the training set.

Keywords: Intrusion Detection System, NSL-KDD dataset, Attribute Selection, Random forest, Classifier

1. Introduction

In the era of information society, computer networks and their related applications are becoming more and more popular, so does the potential threat to the global information infrastructure to increase. To defend against various cyber-attacks and computer viruses, lots of computer security techniques have been intensively studied in the last decade, namely cryptography, firewalls, anomaly and intrusion detection.

Classification of IDSs:
1. **Network intrusion detection systems:** NIDS are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. It performs an analysis of passing traffic on the entire subnet, and matches the traffic that is passed on the subnets to the library of known attacks.

2. **Host intrusion detection systems:** HIDS run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or administrator if suspicious activity is detected.

Apart from these, IDSs are also categorized based on the detection methods as follows:
1. **Signature-based:** They refer to the detection of attacks by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware.
2. **Anomaly-based:** They were primarily introduced to detect unknown attacks, in part due to the rapid development of malware. The basic approach is to use machine learning to create a model of trustworthy activity, and then compare new behavior against this model.

2. Literature Survey

The NSL-KDD data set is the refined version of the KDD cup99 data set. The original DARPA data set is not used because of a number of prominent reasons. The first important deficiency in the KDD data set is the huge number of **redundant records**. The existence of these repeated records in the test set, on the other hand, will cause the evaluation results to be **biased**. Another problem is that the models built **over fit** the data due to which it cannot identify any new attacks that occur.

Prominent **data mining** algorithms like Naïve Bayes, Decision Tree, K-nearest neighbor, K-Means and Fuzzy C-Mean clustering algorithms and **machine learning** strategies like Support Vector Machine (SVM), neural nets have been applied to study the classification.

3. Data set description

This original data set is built based on the data captured in DARPA'98 IDS evaluation program [6]. DARPA'98 is about 4 gigabytes of compressed raw (binary) tcpdump data of 7 weeks of network traffic, which can be processed into about 5 million connection records.

The inherent drawbacks in the KDD cup 99 dataset has been revealed by various statistical analyses has affected the detection accuracy of many IDS modelled by researchers. NSL-KDD data set [3] is a refined version of its predecessor.

In each record there are 41 attributes unfolding different features of the flow and a label assigned to each either as an attack type or as normal.

The attributes can divided into 4 kinds of categories depending upon what it signifies.

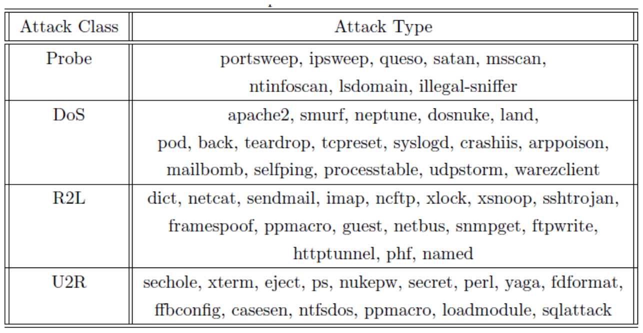
1. **Basic features:** These include attributes which are present in every internet traffic tuple. Example for basic features are Duration, Protocol_type, Service, Flag, Src_bytes, Dst_Bytes, Land, Wrong_fragment, Urgent.
2. **Content related features:** These features are specific to parameters like whether the user is attempting any file accesses or root access, etc. The examples are Hot, Num_failed _logins, Num_comp romised, Logged_in, Root_shell, Su_attempt ed, Num_file_c reations, Is_guest_lo gin, etc
3. **Time related features:** These determine the number of connections which have similar features in the last few seconds. Examples are Count, Srv_count, Serror_rate, Srv_serror_rate,Rerror_rate, Same_srv_rate,etc.
4. **Host based traffic features:** They describe the frequency of queries for same IP, same service, flag, ports, etc. Examples are Dst_host_coun t, Dst_host_same_srv_rate, Dst_host_srv_ diff_host_rate, Dst_host_rerro r_rate, Dst_host_same _src_port_rate, etc

4. Classification of attacks

The data set in KDD have normal and 22 attack type data. These attacks are classified into 4 categories based on the description. All generated traffic patterns end with a label either as 'normal' or any type of 'attack' for upcoming analysis. The 4 main classes of attacks are:

1. **Denial of service (DoS):** Denial of Service is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, denying legitimate users access to a machine.
2. **User to root (U2R):** an attacker starts with access to a normal user account on the system and gains root access. Regular programming mistakes and environment assumption give an attacker the opportunity to exploit the vulnerability of root access.
3. **Remote to user (R2L):** In Remote to User attack, an attacker sends packets to a machine over a network that exploits the machine's vulnerability to gain local access as a user illegally.
4. **Probing:** Probing is a class of attacks where an attacker scans a network to gather information in order to find known vulnerabilities. An attacker with a map of machines and services that are available on a network can manipulate the information to look for exploits.

The mapping of the 22 attacks into these 4 main classes is shown in the table 4.1



Attack Class	Attack Type
Probe	portsweep, ipsweep, queso, satan, msscan, ntinfoscan, lsdomain, illegal-sniffer
DoS	apache2, smurf, neptune, dosnuke, land, pod, back, teardrop, tcpreset, syslogd, crashiis, arppoison, mailbomb, selfping, processtable, udpstorm, warezclient
R2L	dict, netcat, sendmail, imap, ncftp, xlock, xsnoop, sshtrojan, framespoof, ppmacro, guest, netbus, snmpget, ftpwrite, httptunnel, phf, named
U2R	sechole, xterm, eject, ps, nukepw, secret, perl, yaga, fdformat, ffbconfig, casesen, ntfsdos, ppmacro, loadmodule, sqlattack

Table 4: Attacks and classes

5. Experimental results and Analysis

5.1 Setting up workspace and data

The NSL-KDD has a total size of 125000 tuples when compared to the original KDD set which had around 250000 tuples in the 20%subset. The IDE used is R Studio and the classifier models and algorithms are implemented in R as it is easier to analyze and compare the results and do pre processing.

5.2 Pre-processing and Attribute selection step

The only preprocessing step was to label the attacks into 4 main classes and also to normalize some of the attributes to the range of 0-1. The attribute selection is the tricky part. Typical models like random forests, decision trees, etc take very long time to build the model. Hence, the number of attributes must be reduced. Moreover, there are many zero variance features which are removed in this step.

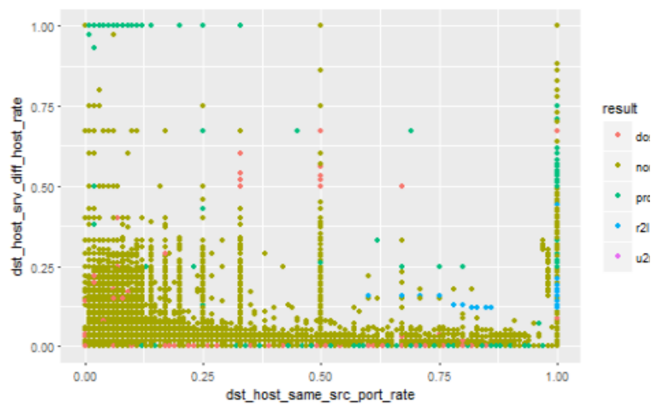


Fig 5.1: Plot of diff_hst_service vs same_host-service_rate

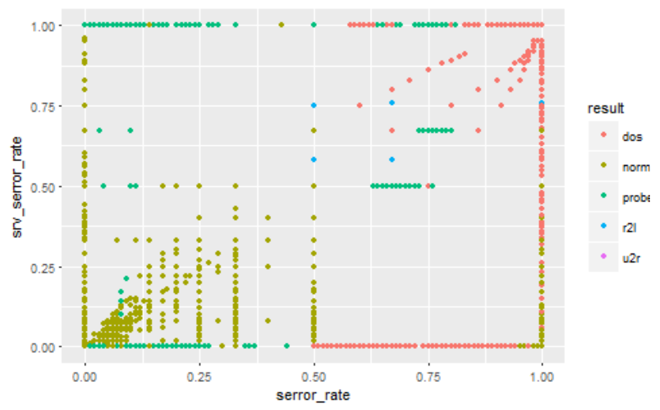


Fig 5.2: Plot srv_error_rate vs serror_rate

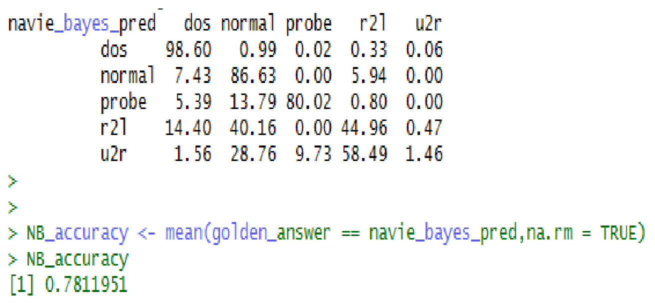
Some plots to illustrate the dependency between the attributes are shown above.

5.3 Building the classifier and testing the results

Three algorithms were used to build the classifier models and each different combinations of number of attributes and the size of training set was used to analyze and compare the efficiencies. The following are the models built and a brief description

5.3.1 Naïve Bayes

This model is based on conditional probability calculations and hence is a mathematical model. The maximum efficiency was obtained when 23 attributes and 125974 training and 25974 testing was used.



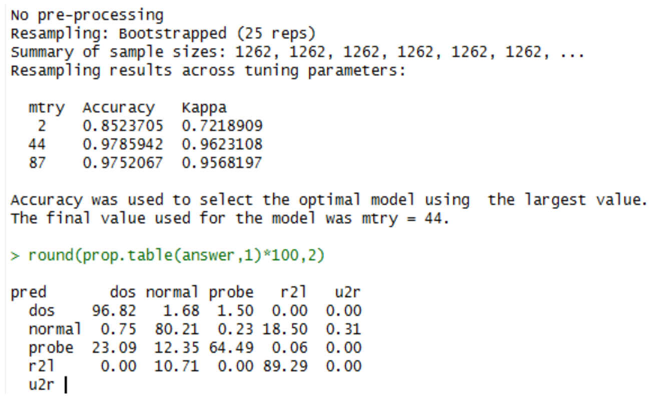
```
navie_bayes_pred   dos normal probe   r2l   u2r
          dos    98.60   0.99  0.02  0.33  0.06
          normal  7.43  86.63  0.00  5.94  0.00
          probe   5.39  13.79 80.02  0.80  0.00
          r2l    14.40  40.16  0.00 44.96  0.47
          u2r     1.56  28.76  9.73 58.49  1.46
>
>
> NB_accuracy <- mean(golden_answer == navie_bayes_pred,na.rm = TRUE)
> NB_accuracy
[1] 0.7811951
```

Fig 5.3: Naïve Bayes model and accuracy

The model gave an average of 78%accuracy for full length of data set.

5.3.2 Random Forest

Random Forests are a group of decision trees built by taking a particular subset of attributes or tuples as input for each tree. Then it uses a bagging and bootstrapping algorithm to ensemble the results obtained. The maximum efficiency was obtained when 23 attributes and 125974 training and 25974 testing was used.



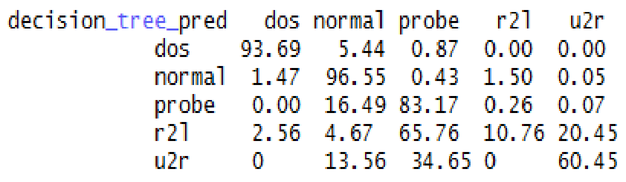
```
No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1262, 1262, 1262, 1262, 1262, 1262, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8523705  0.7218909
  44    0.9785942  0.9623108
  87    0.9752067  0.9568197

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 44.

> round(prop.table(answer,1)*100,2)

pred        dos normal probe   r2l   u2r
  dos     96.82   1.68  1.50  0.00  0.00
  normal   0.75  80.21  0.23 18.50  0.31
  probe   23.09  12.35 64.49  0.06  0.00
  r2l      0.00  10.71  0.00 89.29  0.00
  u2r |
```

Fig 5.4: Random Forests results

5.3.3 Decision Tree

It evaluates feature subsets using best-first search and can use cross-validation for evaluation. The order of attributes and the weight are decided by the algorithm itself.

The maximum efficiency was obtained when 8 attributes and 100000 training and 25974 testing was used.



```
decision_tree_pred   dos normal probe   r2l   u2r
            dos    93.69   5.44  0.87  0.00  0.00
            normal  1.47  96.55  0.43  1.50  0.05
            probe   0.00  16.49 83.17  0.26  0.07
            r2l     2.56   4.67  65.76 10.76 20.45
            u2r     0     13.56  34.65 0     60.45
```

Fig 5.5: Decision Tree efficiency

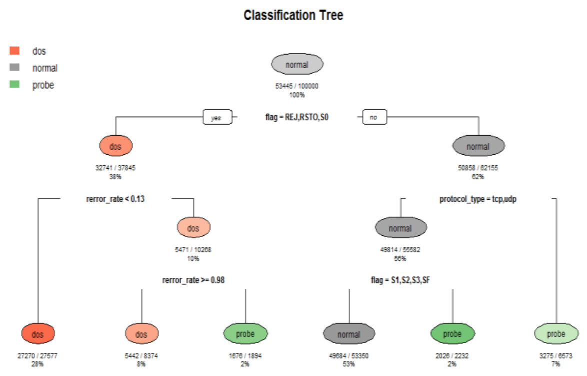


Fig 5.6: Decision Tree model with 5 attributes

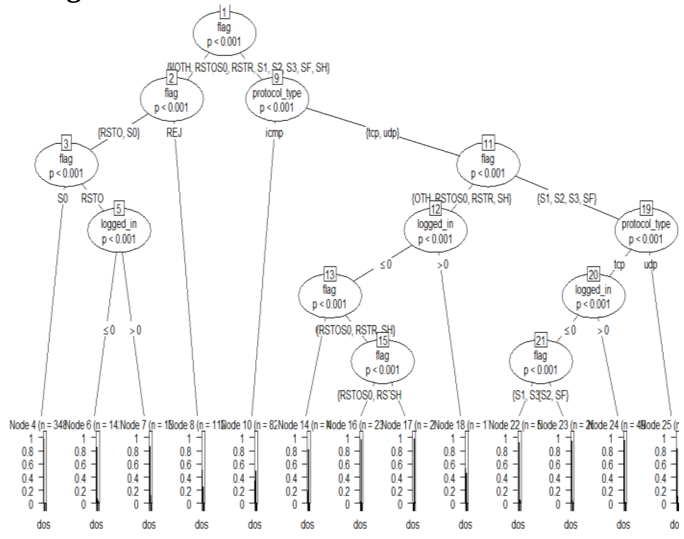


Fig 5.7: Decision Tree model with 8 attributes

6. Conclusions and future work

The pre-processing, construction of classifiers comparison, analysis gave us some insights. Some of them are listed below.

- The redundancy can be eliminated by finding out the zero-variance features and not including them in the training set as it leads to biased results.
- Once threshold efficiency is reached, any increase in the training set would only over-fit the data. Therefore the results would tend to deteriorate if training set is kept increasing.
- Instead the number of attributes taken to build the model can be increased to improve the results
- Once trained, the model was also tested to find out how it would respond to attacks which are new. The models performed pretty well for even these un-trained new attack types.

The future work would be to find out an ensemble way of combining the results of multiple classifiers to find out the best possible prediction. Also, the model must be made robust to new attack types so that it would fail or crash when something for it was trained for tries to intrude the system.

7. References

[1] KDD Cup 1999 Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2] Xu, X.: Adaptive Intrusion Detection Based on Machine Learning: Feature Extraction, Classifier Construction and Sequential Pattern Prediction. International Journal of Web Services Practices 2(1-2), 49–58 (2006)

[3] Vipin Kumar, Himadri Chauhan, Dheeraj Panwar, “K-Means Clustering Approach to Analyze NSL-KDD Intrusion Detection Dataset”, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-4, September 2013

[4] Santosh Kumar Sahu Sauravranjan Sarangi Sanjaya Kumar Jena, “ A Detail Analysis on Intrusion Detection Datasets”, 2014 IEEE International Advance Computing Conference (IACC)

[5] Abdoul Karim Ganame, Julien Bourgeois, Renaud Bidou and Francois Spies, “A global security architecture for intrusion detection on computer networks”, Elsevier, Journal of Computers & Security Vol.27, No.1-2, pp. 30-47, 2008.

[6] Azzedine Boukerche, Kathia Regina Lemos Juca, Joao Bosco Sobral, Mirela Sechi Moretti and Annoni Notare, “An artificial immune based intrusion detection model for computer and telecommunication systems”, Elsevier, Journal of Parallel Computing, Vol.30, No.5-6, pp. 629–646, 2004.

[7] Abdun Naser Mahmood, Christopher Leckie, and Parampalli Udaya, “An Efficient Clustering Scheme to Exploit Hierarchical Data in Network Traffic Analysis”, IEEE Trans. on Knowledge and Data Engineering, Vol. 20, No. 6, pp. 752-767, 2008.

[8] Ajith Abraham, Ravi Jain, Johnson Thomas and Sang Yong Han, “D-SCIDS: Distributed soft computing intrusion detection system”, Elsevier, Journal of Network and Computer Applications, Vol.30, No.1, pp. 81–98, 2007..

[9] A. K. Jain, S. Pankanti, S. Prabhakar, L. Hong, and A. Ross, “Biometrics: a grand challenge”, Proc. Int’l Conf. on Pattern Recognition, vol. 2, pp. 935–942, August 2004.

[10] J. Leggett and G. Williams, “Verifying Identity via Keystroke Characteristics”, Int’l J. Man-Machine Studies, vol. 28, no. 1, pp. 67–76, 1988.

[11] Y. Li, B. Zhang, Y. Cao, S. Zhao, Y. Gao and J. Liu, “Study on the Beihang Keystroke Dynamics Database”, Int’l Joint Conf. on Biometrics (IJCB), pp. 1-5, 2011.

[12 Charles R. Haag, Gary B. Lamont, Paul D. Williams and Gilbert L. Peterson, “An Artificial Immune System-Inspired Multi objective Evolutionary Algorithm with Application to the Detection of Distributed Computer Network Intrusions”, Proc. of ACM conference GECCO’07, July 7-11, pp. 2717-2724, 2007.

[13] S. T. Brugger, J. Chow, An assessment of the DARPA IDS evaluation dataset using Snort, Tech. Report, CSE-2007-1, Nov. 2005

[14 W. Lee, S.J.Stolfo, A Data Mining framework for building intrusion detection models, IEEE Symposium on Security and Privacy, 1999

[15] J. McHugh, A. Christie, J.Allen, Defending Yourself: The Role of Intrusion Detection Systems, IEEE software, Sep/Oct. 2000

[16] S. Haider, A. Abbas, and A. K. Zaidi. A multi-technique approach for user identification through keystroke dynamics. IEEE International Conference on Systems, Man and Cybernetics, pages 1336–1341, 2000.

[17] H.-j. Lee and S. Cho. Retraining a keystroke dynamicsbased authenticator with impostor patterns. Computers & Security, 26(4):300–310, 2007.

[18] Zhan Jiuhua, “Intrusion Detection System Based on Data Mining”, Proc. of IEEE Workshop on Knowledge Discovery and Data Mining (WKDD’08), pp. 402-405, 2008.

[19] Young U. Ryu and Hyeun-Suk Rhee, “Evaluation of Intrusion Detection Systems under a Resource Constraint”, ACM Trans. on Information and Systems Security, Vol. 11, No. 4, pp. 20:1-20:23, 2008.

[20] Vladimir Golovko and Pavel Kochurko, “Intrusion Recognition Using Neural Networks”, IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, (IDAACS’05), pp. 108-111, 2005.