- **Title:**

Optimizing Neural Network Architecture Using Tikhonov Regularization Parameter For Intrusion Detection Systems and Classification of Attacks.

- **Names:**

Marwa S.Sharawi[1], Mohammed Sammany[2], Mohammed El-Beltagy[3], Imane Saroit[4]

Affiliations:

- The British University in Egypt, Egypt
- Cairo University, Egypt
- Aleppo University, Syria

- **Postal Address:**

6 al obour Building, floor 14, flat 3, Salah Salem street, heliopolis , Cairo, Egypt.

- **Email Addresses and phones:**

Marwa Sa`id Sharawi E-mail: mssharawi@hotmail.com msaid@Bue.edu.eg Mobile: 002 012 484 25 65	Mohammed El-Beltagy E-mail: Mohammed @ computer.org Mobile: 002 010 191 66 50
Imane Aly Saroit E-mail: iasi63@hotmail.com Mobile: 002 012 245 24 79	Mohamed Sammany E-mail: sammanyddr1@hotmail.com Mobile: 00963-0947-742552

- **Contact Author:**

Mss. Marwa Sa`id Sharawi
E-mail: mssharawi@hotmail.com
msaid@Bue.edu.eg
Mobile: 002 012 484 25 65

- **Key Words:**

Intrusion Detection Systems, Neural networks, Multi-layered Perceptron (MLP), Tikhonov Regularization Parameter.

- **Conference Name:** ***IKE`08*** (The 2008 International Conference on Information and Knowledge Engineering)

Optimizing Neural Network Architecture Using Tikhonov Regularization Parameter For Intrusion Detection Systems and Classification of Attacks

Marwa S.Sharawi[1], Mohammed Sammany[2], Mohammed El-Beltagy[3], Imane Saroit[4]

***Abstract*—Network Intrusion Detection Systems (NIDS) require the ability to generalize from previously observed attacks to detect even new or slight variation records of known attacks. As an intrusion detection system can be regarded as classification problem, we use Artificial Neural networks for detection. Using a benchmark study and set from the KDD (Knowledge Data Discovery and Data Mining) competition designed by DARPA and Multi-layered perceptron neural network, this Paper will aim to solve a multi class problem using MLP in to distinguish the attack records from normal ones, and also identify the attack type. In addition, it shows how to use Tikhonov regularization parameter to optimize the optimal network architecture in order to increase the system performance. The results show that the designed system is capable of classifying records with 98.34% accuracy with two hidden layers of neuron. Finally, the performance of the benchmark study is compared with our results.**

***Index Terms*— Intrusion Detection Systems, Neural networks, Multi-layered Perceptron (MLP), Tikhonov Regularization Parameter.**

M.S.Sharawi, Faculty of Informatics and Computer Science, British University in Egypt, Egypt.
(E-mail: mssharawi@hotmail.com, msaid@Bue.edu.eg).

M.Sammany, Department of Applied Mathematics and Programming, Aleppo University, Syria.
(e-mail: sammanyddr1@hotmail.com).

M. El-Beltagy., Department of Decision Support System, Faculty of Computers and Information, Cairo University, Egypt.
(E-mail: Mohammed @ computer.org).

I. Saroit, Department of Information Technology, Faculty of Computers and Information, Cairo University, Egypt.
(e-mail: iasi63@hotmail.com).

I. Introduction

As a result of the great evolution and expansion in the world of the web and local network systems, the central concern focus on the possibility to prevent and detect all possible attacks to the systems and the ability to reduce the cost of the great damaged caused by the unauthorized access to these systems. The Highly connected computing world equipped the intruders and hackers with new facilities for their destructive purposes and this leads to the need of detection systems that minimize the attack process [1]. There are two main approaches to the design of IDSs, that is, *misuse* and *anomaly* detection techniques [2]. In a misuse detection based IDS, intrusions are detected by looking for activities that correspond to known signatures of intrusions or vulnerabilities. Otherwise, anomaly detection based IDSs depend on the variation of the normal network traffic. The abnormal traffic pattern can be defined either as the violation of accepted thresholds for frequency of events in a connection or as a user's violation of the legitimate profile developed for normal behavior.

The first and most well known intrusion detection system was Denning's system which is expert system based on rule-based analysis using Denning's profile model [3]. It is depending on sets of predefined rules that are previously provided by an administrator or created by the system.

Unfortunately, these expert systems require frequent updates to remain up to date. This design approach usually leads to an inflexible detection system that is unable to detect an attack if the sequence of events is even slightly different from the predefined profile. The problem comes when the intruder is an intelligent and flexible agent while the rule based IDSs obey fixed rules [4]. To overcome this problem, some optimization techniques that are tolerant of imprecision and uncertainty are required. ANN (Artificial Neural Network) regards as one of the most powerful techniques in learning and generalization capabilities for implementing IDSs. It is capable to classifying the novel attacks from normal connection audit records, and to generalize the patterns to new (slightly different) data record of the same class. In fact, there has been a huge literature achieved to address the intrusion detection problem using neural networks. However, this paper aimed to solve an off-line multi class problem based on the idea of Moradi and

Zulkernine [4] as a benchmark study for our research. Here we are focus to the answer what mentioned in [4].

"The question is if anything is gained by using more than one hidden layer. One answer is that using more than one layer may lead to more efficient approximation or to achieving the same accuracy with fewer neurons in the neural network."

We will mainly stress in using the same type of data and classification in order to be able to compare the performance results to our proposed model. We will work on the way of using different architecture structure of the MLP neural network as an intrusion classifier in order to reserve the style of our benchmark study and to find out the relation between the best performance and the network architecture. Then we will solve the problem of choosing the optimal neural network architecture to ensure the highest classification result; this will leads to use a tikhonov regularization parameter as a solution of an ill-posed problem. The [4] study was aimed to solve a multi class problem. There are, a three-class case is described which can be extended to cases with more attack types. An output layer with three neurons (output states) was used: [1 0 0] for normal conditions, [0 1 0] for Neptune attack and [0 0 1] for the Satan attack. The desired output vectors used in training, validation, and testing phases were simply as mentioned above. The [4] study in which not only the attack records are distinguished from normal ones, but also the attack type is identified. However, the total average accuracy obtained by their proposed neural network model was 90%, using the early stopping criteria as method to stop the training. In this study, we based on the regularization theory of Tikhonov [5] to optimize the neural network architecture in order to improve the system performance and maximize the generalization capability. Applying the regularization theory of Tikhonov in the back-propagation algorithm leads to emerging other stopping points that are, noteworthy, better to stop our training process than the early stopping criteria used in [5]. The optimal value of Tikhonov stabilizer has been calculated according to Rögnvaldsson [6]. This feature of Tikhonov regularizer enables the system to reach 98.3% of correct classification accuracy against possible attacks. The promising results of the present study show the potential applicability of ANNs for developing high efficiency practical IDSs.

Paper Organization: - Section I has introduced the basic ideas of intrusion detection and the motivations for this study. Section II introduce the basic ideas about the intrusion detection, kinds of intrusion detection Systems, types of attacks, and the features used for classifying network connection records. Section III presents the neural network method as an intrusion detector. Section IV describes theory of optimizing the network architecture using Tikhonov regularization as a solution of ill-posed problem. In Section V, we present the implementation procedure and the discussion of our experimental results. Finally, the paper ends with a conclusion and the possibilities for future work.

II. Intrusion Detection System

A. An Overview

Intrusion detection is software, hardware or combination of both used to detect intruder activity by using a set of techniques and methods at the network and host level. There are two types of intrusion detection systems, that is, the *Network Intrusion Detection Systems* (NIDS) and the Host Intrusion Detection Systems (HIDS) [7]. NIDS are intrusion detection systems that capture data packets traveling on the network media and match them to a database of signatures, which are the pattern that we look for inside data packets. Depending upon whether a packet matched with an intruder signature, an alert generated. However, the HIDS installed as agents on a host. These intrusion detection systems can look into system and application log files to detect any intruder activity and alert it. These alerts are any sort of user notification such as, pop-up windows, logging to a console, sending e-mail and so on. Generally, Intrusion detection systems fall into three basic categories [8]:

- **Signature-based intrusion detection systems**
  These systems based upon a set of predefined signatures and rules. These detection process occur when a comparison process take place and found a matched signature patterned and then generate alerts.

- **Anomaly-based intrusion detection systems**
  It is also known as profile-based detection since the systems purpose is to examine the statistical and characteristic behavior. It depends on the normal behavior of the system and indicates an intruder depending on the variation of the normal behavior.

There are four phases of the analysis process applied in the both types of detection system [8]. These phases are *preprocessing*, *analysis*, *response***,** and *refinement*.

Preprocessing is a process, which contains different classifications processes, with purpose to format the data collected from IDS. Formatted data can be easily broken down further into classifications. Those classifications depend on analysis methods we use, i.e. rule-based or profile-based detection. ***Analysis*** phase is done through a comparison between knowledge base and classified data. The data can logged as intrusion in this phase. ***Response*** can be performed manually after analysis or automatically under present attack, it consists of few activities such are: alert sending via SMS, mail, SNMP trap etc. ***Refinement*** phase we try to fine-tune our IDS policy, which gives us opportunity to minimize number of false-positives.

B. Types of Networking Attacks

In the world of intrusion detection, there are at least four major categories of networking attacks. Every attack on a network can be placed into one of these groupings (Denial of service (DOS), Remote to User attack (R2U), User to Root attack (U2R) and probe attack) [9].

From the previous mentioned four different attack types this study will include only two types: *SYN Flood (Neptune)* and *Satan.* These two attack types were selected from two different attack categories (denial of service and probing) in order to test and validate the ability of the proposed intrusion detection system to identify attacks from different categories. As the benchmark study used the same attack types [10], evaluation of the results by comparing them to previous studies was possible. In the following paragraphs, a description of the attack types is provided [4].

- *SYN Flood (Neptune)* is a denial of service. Network traffic is monitored for a number of simultaneous SYN packets destined for a particular machine attack to which every TCP/IP implementation is possibly vulnerable.

- *Satan* is a probing intrusion, which scans a network of computers to collect information or discover known vulnerabilities.

The purpose of classifiers in IDSs is to identify attacks from all four groups as accurately as possible.

III. Data Sets and Features

The 1999 standard version of MIT Lincoln Laboratory – DARPA (Defense Advanced Research Projects Agency) intrusion detection data was used in this research [4,10]. In DARPA dataset, each event (connection) is described with 41 features. 22 of these features describe the connection itself and 19 of them describe the properties of connections to the same host in last two seconds. In many attack scenarios, the signature of the attack record is identified through examination of some features in a sequence of records. Therefore, the IDS should analyze the service types used by the same user in previous connections and for this purpose these 19 features describing past events in the computer network were included in the feature vector. According to [4] "The sample version of the dataset included more than 450,000 connection records. A subset of the data that contained the desired attack types and a reasonable number of normal events were selected manually..." The final dataset used in this study included 13000 records (the same used in [4]), and the data vector was a *35 dimensional* vector. A complete description of all 41 features is available [4, 9]. A complete description of the feature vector will be in Appendix A. *"Instead of describing all the features, here we divide them into three groups and provide descriptions and examples for each group"* [4].

- **Group 1** includes *commands* features used in the connection. It describes the aspects of the commands that have a key role in defining the attack scenarios. Examples of this group are number of file creations, number of operations on access control files, number of root accesses, etc.
- **Group 2** includes *connection specifications* features. It presents the technical aspects of the connection. Examples of this group include, protocol type, flags, duration, service types, and number of data bytes from source to destination, etc.
- **Group 3** includes *connections to the same host in last 2 seconds* features. Examples of this group are, number of connections having the same destination host and using the same service, number of connections to the current host that have a rejection error, number of different services on the current host, etc.

IV. Artificial Neural Networks and Intrusion Detection

Neural networks are a uniquely powerful tool in multiple class classification [11], especially when used in applications where formal analysis would be very difficult or even impossible, such as pattern recognition, nonlinear system identification, and control. Provided the neural network has been given sufficient time to train, the property of generalization ensures that the network will be able to classify patterns that have never been seen before. However, the accuracy of classification problems depends on a variety of parameters, ranging from the architecture of the actual neural network to the training algorithm of choice.

MLP is a multiplayer feed-forward network consists of an input layer, one or more hidden layers, and an output layer. The output layer supplies the response of the network to the activation patterns applied to the input layer. The present study is aimed to solve a three-class class problem, which can be extended to cases with more attack types. The objective of MLP is to assign the input patterns to one of the categories that are represented in terms of neural networks' outputs, so that they represent the probability of class membership. The symbolic representation has been used to express each of the three conditions in such a way that, a "1" in a column indicates the occurrence of the column's corresponding string and a "0" indicates a non-occurrence. Thus, we have three cases of classification probabilities, that is, $[1\ 0\ 0]$ for Normal conditions, $[0\ 1\ 0]$ for Neptune attack and $[0\ 0\ 1]$ for the Satan attack exactly as used in the [10] study. The *TanhSigmoid* activation functions will use for every neuron. In [4] study the main problem occurred during neural network training was the over-fitting.

A. The Over-fitting Problem

In an over-fitted ANN, the error (number of incorrectly classified patterns) on the training set is driven to a very small value, however, when a new data is presented, the error is large. In these cases, the ANN has memorized the training examples; however, it has not learnt to generalize the solution to new situations. One possible solution for the over-fitting problem is an improving generalization is called *early stopping criteria* [12]. In this technique, the available data is divided into three subsets. The first subset is the training set, which is used for training and updating the ANN parameters. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phase of training similar to the training set error. However, when the ANN begins to over-fit the data, the error on the validation set will typically begin to rise.

When the validation error increases for a specified number of iterations, the training is stopped, and the weights that

produced the minimum error on the validation set are retrieved [13]. Early stopping has the advantage of being quick, since it shortens the training time. However, it has the disadvantages of being poorly defined (does not guarantee the maximum generalization) and not making full use of the available data [6]. Since our objective is to achieve the maximum generalization with the minimum neural network structure, we used *regularization theory of Tikhonov* in training our neural network model. A brief description of the Tikhonov regularization method and the estimate used to determine the *Tikhonov regularization parameter* [14] (in some literatures called Tikhonov stabilizer, Tikhonov Parameter) is given in the next section.

V. Optimizing Neural Networks Architecture Using Tikhonov Regularization Parameter As a Solution of Ill-posed Problem

Regression problems, which do not put constraints on the model used is ill-posed [6], because there are infinitely many functions that can map a finite set of training data perfectly. Furthermore, raw data sets tend to have noisy inputs and/or outputs, which is why models that fit the data perfectly tend to be poor in terms of out-of-sample performance. Since the modeler's task is to find a model for the underlying function while not over-fitting to the noise. An effective way to control the neural network performance is to reduce the size of the network i.e. eliminating (or decaying) the superfluous connections (weights) among the neurons. In fact, this can be done by creating a driving force that attempts to decrease all the weights to zero during adaptation process. If the input-output map requires some large weights, learning will keep bumping up the important weights, and the ones that are not important will be driven to zero, thus reducing the number of free parameters (degrees of freedom) in the network. This process referred to as Tikhonov regularization. The key idea of regularization is "*imposing a penalty term in the performance function*" which can be implemented very simply by adding an extra term into the weight adaptation, as shown here for the *gradient descent rule* [15].

$$w_{ji}(n+1) = w_{ji}(n)\left(1 - \frac{\lambda}{(1+w_{ji}(n)^2)^2}\right) + \alpha(\Delta w_{ji}(n)) + \eta\delta_i(n)x_j(n) \quad (1)$$

where δ is the local error, α is the momentum constant, η the learning rate, and λ is called Tikhonov regularizer, which can be estimated, optimally, by the equation:

$$\hat{\lambda} = \frac{\left\|\nabla E(W_{es})\right\|}{\left\|2 W_{es}\right\|} \quad (2)$$

Where W_{es} is the set of weights at the early stopping point, and $E(W)$ is the training error.

VI. Experimental Results and Classifier Evaluation

Using *Back-propagation algorithm* [16], a two hidden layers MLP with 50 and 30 neurons, in the first and the second layer respectively, has been implemented as an intrusion detector. The learning rate has been chosen in such a way that it has high value between the input and hidden layers, while they have a small values between the second and the output layer. This is due to the fact that the high values of learning rates in the backward layers speed up the convergence rate without affecting the stability of the network [17]. Using E.q(2) the Tikhonov parameter has been calculated, and the neural network has been re-trained again using equation (1). The proposed model has been trained on a training set consists of 10400 patterns, twenty percent of the training set has been truncated for validation, and the network has been tested on 2600 pattern not seen before (apportioned equivalently upon the three classes). Fig. 1(A and B) reveals the mean square error of the training process versus the progress of training epochs for the training and validation sets, and the stopping criterion of the training, with on without the presence of the Tikhonov parameter. Table 1 shows the number of patterns that are correctly classified and rate of correct classification for each class in addition to the average classification accuracy of the three classes, which measures the whole performance of the proposed model.

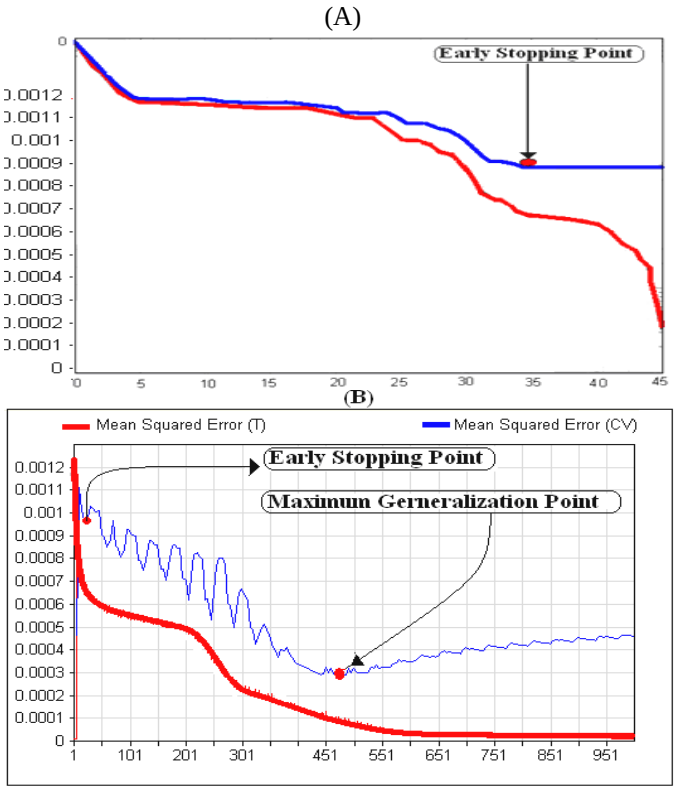

Figure 1. Mean square error of the training process versus the progress of training epochs for the training and validation sets. (A) Without including the Tikhnov Regularizer [11], (B) Using Tikhonov Regularizer $\lambda = 1 \times 10^{-7}$

Table 1 Testing and Classification

Class Name	Number of Test Patterns for each Class	Number of Correctly Classified Patterns	Correct Classification
Normal	866	857	98.96 %
Neptune	866	831	95.95 %
Satan	866	866	100 %
		AVR	98.34%

Discussion

- At the first stage of our experiment, early stopping criterion for validation set was applied to stop the training process. At this point, the number of epochs was 35 and the value of Cross Validation Error (CVE) was 0.0014. Then, the training has been continued again and the weights have been saved for each epoch until the training process has been stopped after 1000 epochs. While monitoring the CVE along with the MSE for the training process, it has been noticed that the CVE still decreasing, wiggly, until it reached a global minimum, where the number of epochs was 475 and the CVE was 0.0003. Starting from this point, the CVE started to rise again; therefore, this point can be considered the maximum generalization point.

- Using the saved weights, the network has been tested on the second stopping stage, and compared to the first stopping point. As expected, the correct classification rate on the training set declined slightly (from 0.0014 to 0.0003). Instead, when unseen data (test set) was fed to the neural network the result was better than the first stage in which the early stopping method was applied. The correct classification rate was more than 98.3% showing an 8.3 % increase (from 90% in the first stage, the results of [10]).

VII. Conclusion and Future Work

In this paper we present an intrusion detection system based neural networks. The implemented neural network model used to solve a three-class problem, that is, normal, attack patterns, and the type of the attack. However, its further development to several classes is straightforward. Tikhonov Regularization Parameter has been applied during the weight adjustment process. Adding the Tikhonov parameter creates a driving force that attempts to decrease all the weights to zero during adaptation process, thus reducing the number of free parameters (degrees of freedom) in the network, and hence optimize the network architecture. We showed that regularization using Tikhonov Parameter might lead to more points that are important to stop the training. Thus, enhance the chance of reaching the global minima, and increase the generalization capability. When unseen data is presented to the model, the results obtained revealed a great deal of accuracy (98.30%) in comparison with other studies, that depends on the early stopping criteria, only, to stop the training.

Practical IDSs should have several attack types; therefore, it is possible, as a future development to the present study, to include more attack scenarios in the dataset. In addition, in order to avoid unreasonable complexity in the neural network, an initial reduction of the records to normal and general categories of attacks can be done as a the first step, before introducing the data to the network.

Acknowledgment

The author would like to thank:

- Mehdi Moradi, from Queen's University Kingston, Ontario, Canada, for presenting his work in this domain and for giving the data sets used in our experimentations, and for his guidance while doing this research.
- Thorsteinn Rögnvaldsson from Halmstad University, Halmstad, Sweden, for his guidance and support regarding the regularization theory.

References

[1] R. A. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," Computer, vol. 35, no. 4, pp. 27–30, 2002.

[2] W. Lee, S. J. Stolfo, K. Mok, "A Data Mining Framework for Building Intrusion Detection Models", *Proceedings of 1999 IEEE Symposium of Security and Privacy*, pp. 120-132.

[3] D. E. Denning, "An intrusion detection model," IEEE Transactions on Software Engi neering, vol. 13, no. 2, pp. 222– 232, 1987.

[4] Mehdi MORADI and Mohammad ZULKERNINE, " A Neural Network Based System for Intrusion Detection and Classification of Attacks,"*Proceedings of the 2004 IEEE World Congress on Computational Intelligence*, Honolulu, HI, pp. 1714-1719, 2004.

[5] A.N. Tikhonov and V. Y. Arsenin. " Solutions of Ill-Posed Problems," V.H.Winston & Sons, Washington D.C., 1977.

[6] T.S. Rögnvaldsson. "A Simple Trick for Estimating the Weight Decay Parameter", Springer Verlag 1998.

[7] Rafeeq Ur Rehman,Intrusion Detection Systems with snort, Prentice Hall PTR Upper Saddle River, New Jersey 07458, www.phptr.com

[8] Carl Endorf, Eugene Schultz & Jim Mellander (2004). Intrusion Detection & Prevention. The McGraw-Hill/Osborne Companies.

[9] Das, K., J., "Attack Development for Intrusion Detection Evaluation,", *Master thesis, Massachusetts Institute of Technology (MIT), USA*, 2000.

[10] MIT Lincoln Laboratory, http://www.ll.mit.edu.

[11] Theodorios and Konstantinos Koutroumbas, *Pattern Recognition*, Cambridge: Academic Press, 1999.

[12] S.Haykin, Neural Network Comprehensive Foundation, Second Edition , Prentice Hall PTR Upper Saddle River, New Jersey 07458, www.phptr.com

[13] MATLAB online support: www.mathworks.com/access/helpdesk/help/techdoc/matlab.sht ml.

[14] Tikhonov, A.N.,"On regularization of Ill-posed Problems and Method of Regularization." Doklady Akademii Nauk USSR, vol.151,pp.501-504. (1973).

[15] Principe, J.C. "Artificial Neural Networks" *The Electrical Engineering Handbook* Ed. Richard C. Dorf Boca Raton: CRC Press LLC, (2000).

[16] Rumelhart, D.E., Widrow, B., and Lehr, M.A. - "The Basic Ideas in Neural Networks", Communications *of the Association for Computing Machinery*, Vol. 37, No. 3,pp. 87 92.(1994)

[17] Jacques Hadamard " Sur les problemes aux derivees partielles et leur signification physique" (1902).

Appendix A

The feature vector file:

1) Duration: length (number of seconds) of the connection (Numerical, Continuous)
2) protocol_type: type of the protocol (tcp=0, udp=1, icmp=2)
3) service: network service on the destination, e.g., http, telnet, etc. (http=0, ftp=1, private=2, domain_u=3, smtp=4, finger=5, ftp_data=6, ecr_i=7, telnet=8, urp_i=9, auth=10, eco_i=11, ntp_u=12, nntp=13, uucp_path=14, netbios_dgm=15, imap4=16, sql_net=17, vmnet=18, bgp=19, Z39_50=20, exec=21, login=22, shell=23, printer=24, efs=25, courier=26, ldap=27, sunrpc=28, http_443=29, klogin=30, kshell=31, IRC=32, X11=33, discard=34, pop_3=35, netbios_ns =36, pop_2=37, link=38, echo=39, systat=40, daytime=41, netstat=42, ssh=43, time=44, whois=45, name=46, domain=47, mtp=48, gopher=49, remote_job=50, rje=51, ctf=52, supdup=53, iso_tsap=54, hostnames=55, csnet_ns=56, netbios_ssn=57, nnsp=58, uucp=59, other=60 ,
4) flag: normal or error status of the connection (S0=0, SF=1, S1=2, REJ=3, S2=4)
5) src_bytes: number of data bytes from source to destination (Numerical, Continuous)
6) dst_bytes: number of data bytes from destination to source (Numerical, Continuous)
7) land: 1 if connection is from/to the same host/port; 0 otherwise
8)wrong_fragment: number of "wrong" fragments (Numerical, Continuous)
9) urgent: number of urgent packets (Numerical, Continuous)
10) hot: number of "hot" indicators (Numerical, Continuous)
11)num_failed_logins: number of failed login attempts (Numerical, Continuous)
12) logged_in: 1 if successfully logged in; 0 otherwise .
13)num_compromised: number of "compromised" conditions (Numerical, Continuous).
14) root_shell: 1 if root shell is obtained; 0 otherwise .
15)su_attempted 1 if "su root" command attempted; 0 otherwise .
16)num_root: number of "root" accesses (Numerical, Continuous).
17)num_file_creations: number of file creation operations (Numerical, Continuous).
18)num_shells: number of shell prompts (Numerical, Continuous).
19) num_access_files: number of operations on access control files (Numerical, Continuous).
20) num_outbound_cmds: number of outbound commands in an ftp session (Numerical, Continuous).
21) is_host_login: 1 if the login belongs to the "hot" list; 0 otherwise .
22) is_guest_login: 1 if the login is a "guest" login; 0 otherwise .
23)count: number of connections to the same host as the current connection in the past two seconds (Numerical, Continuous).
Note: The following features refer to these same-host connections.
24) srv_count: number of connections to the same service as the current connection in the past two seconds (Numerical, Continuous).
25) serror_rate: % of connections that have "SYN" (S0) errors (Numerical, Continuous).
26) srv_serror_rate: % of connections that have "SYN" errors (Numerical, Continuous).
27) rerror_rate: % of connections that have "REJ" errors (Numerical, Continuous).
28) srv_rerror_rate: % of connections that have "REJ" errors (Numerical, Continuous).
29) same_srv_rate: % of connections to the same service (Numerical, Continuous).
30) diff_srv_rate: % of connections to different services (Numerical, Continuous).
31) srv_diff_host_rate: % of connections to different hosts (Numerical, Continuous).
32) dst_host_count: number of connections having the same destination host (Numerical, Continuous).
33) dst_host_srv_count: number of connections having the same destination host and using the same service (Numerical, Continuous).
34) dst_host_same_srv_rate: % of connections having the same destination host and using the same service (Numerical, Continuous).
35) dst_host_diff_srv_rate: % of different services on the current host (Numerical, Continuous).
36) dst_host_same_src_port_rate: % of connections to the current host having the same src port (Numerical, Continuous).
37) dst_host_srv_diff_host_rate: % of connections to the same service coming from different hosts (Numerical, Continuous).
38) dst_host_serror_rate: % connections to the current host that have an "SYN" error (Numerical, Continuous).
39) dst_host_srv_serror_rate: % of connections to the current host and specified service that have an "SYN"error (Numerical, Continuous).
40) dst_host_rerror_rate: % of connections to the current host that have an "REJ" error (Numerical, Continuous).
41) dst_host_srv_rerror_rate: % of connections to the current host and specified service that have an "REJ" error (Numerical, Continuous).

Target Output: label of the sample connection (normal=0, Neptune=1, Satan=2).