Journal of Information Security, 2012, 3, 259-271
http://dx.doi.org/10.4236/jis.2012.34033 Published Online October 2012 (http://www.SciRP.org/journal/jis)

Category-Based Intrusion Detection Using PCA

Gholam Reza Zargar[1], Tania Baghaie[2]
[1]GIS Department, Khuzestan Electrical Power Distributed Company, Ahvaz, Iran
[2]Training Center of Applied Science and Technology, Tehran Municipality Information and Communication Technology Organization, Tehran, Iran
Email: zargar@vu.iust.ac.ir, baghaie@vu.iust.ac.ir

Received June 16, 2011; revised August 22, 2012; accepted September 5, 2012

ABSTRACT

Existing Intrusion Detection Systems (IDS) examine all the network features to detect intrusion or misuse patterns. In feature-based intrusion detection, some selected features may found to be redundant, useless or less important than the rest. This paper proposes a category-based selection of effective parameters for intrusion detection using Principal Components Analysis (PCA). In this paper, 32 basic features from TCP/IP header, and 116 derived features from TCP dump are selected in a network traffic dataset. Attacks are categorized in four groups, Denial of Service (DoS), Remote to User attack (R2L), Remote to User attack (U2R) and Probing attack. TCP dump from DARPA 1998 dataset is used in the experiments as the selected dataset. PCA method is used to determine an optimal feature set to make the detection process faster. Experimental results show that feature reduction can improve detection rate for the category-based detection approach while maintaining the detection accuracy within an acceptable range. In this paper KNN classification method is used for the classification of the attacks. Experimental results show that feature reduction will significantly speed up the train and the testing periods for identification of the intrusion attempts.

Keywords: Intrusion Detection; Principal Components Analysis; Data Dimension Reduction; Feature Selection; Classification

1. Introduction

Intrusion Detection Systems (IDS) is designed to complement other security measures based on attack prevention (firewalls, antivirus, etc.). Amparo Alonso-Betanzos *et al.* [1] say that "The aim of the IDS is to inform the system administrator of any suspicious activities and to recommend specific actions to prevent or stop the intrusion". Intrusion can be defined as an attempt to gain unauthorized access to network resources [2]. IDS is necessary for effective computer system protection. There are two approaches for intrusion detection, *i.e.* signature-based and anomaly-based intrusion detection. In signature-based or misuse detection method, patterns of well known attacks are used to identify intrusions [3]. In anomaly-based intrusion detection, network traffic is monitored and compared versus any deviation from the established normal usage patterns to determine whether the current state of the network is anomalous. An anomalous traffic can be flagged as intrusion attempt. Misuse detection uses well defined patterns known as signatures of the attacks. Anomaly-based detection builds a normal profile and anomalous traffic is detected when the deviation from the normal model reaches a preset threshold [4].

Signature-based IDSs typically require human input to create attack signatures or to determine effective models for the normal behavior [4]. Feature selection ranking can be used in anomaly-based and signature-based intrusion detection systems. Feature selection is an important issue in intrusion detection. The reason for it is due to the large number of features that should be monitored for the intrusion detection purpose. Elimination of useless or less relevant features will maintain accuracy of the detection while speeding up its calculations. Therefore, any reduction in the number of features used for the detection will significantly improve the overall performance of the IDS. In cases where there are no useless features, concentrating on the most important ones is expected to improve the execution speed of an IDS. This increase in the detection speed will not affect accuracy of the detection in a significant way.

Incorrect selection of the features may not only reduce the speed of the operation but may also reduce detection accuracy [5].

This paper reports a work aimed on improving the intrusion detection time using a category-based intrusion detection model. In **Figure 1**, network traffic in divided into six groups, normal, DoS, R2L, U2R, Probing and Undetermined Anomalous Behavior (UAB). The main goal in a Category-Based Intrusion Detection (CBID) is

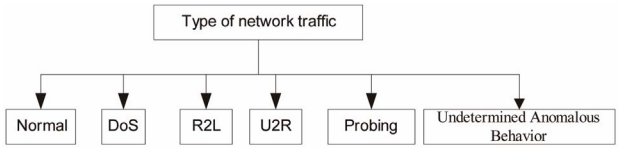

Figure 1. Category-based separation of the network traffic.

to reduce the amount of data that is less important with regard to the intrusion detection and to eliminate them.

This approach has the benefit of reducing memory requirements for storage, reducing data transfer and processing time, and improving the detection rate [6]. IDS has to examine a very large audit data in a short period of time. Therefore, any reduction in the volume of data may save the processing time [7].

Considering certain attack categorizes, some features in the traffic data are more relevant than the rest for intrusion detection.

Feature reduction can be performed in several ways [7-10]. In this paper, the category-based approach is used to find the relevance between features extracted from the network traffic. This paper also proposes a method based on TCP/IP header parameters and derived features selected from TCP dump network traffic dataset. In the proposed approach, Principal Components Analysis (PCA) is used as a dimension reduction technique. KNN classification method is used the detection of the intrusion attempts and results are reported.

2. Related Works

In a reported work, Chakraborty [11] reports that the existence of irrelevant and redundant features generally affects the performance of machine learning part of the work. Chakraborty Proves that proper selection of the feature set results in better classification performance. A. H. Sung *et al.* [8] have demonstrated that the elimination of these unimportant and irrelevant features did not significantly reduced performance of the IDS.

Chebrolu *et al.* [7], report that an important advantage for combining redundant and complementary classifiers is to increase robustness, accuracy and better overall generalization. Chebrolu *et al.* [7] have also identified important input features in building IDS that are computationally efficient and effective. In their reported work, they have investigated performance of three feature selection algorithms, *i.e*. Bayesian networks (BN), Classification and Regression Trees (CART) and an ensemble of BN and CART.

Sung and Mukkamala [8], have explored SVM and Neural Networks to identify and categorize features with respect to their importance to detect specific kinds of attacks such as probing, DoS, Remote to Local (R2L), and User to Root (U2R). They have also demonstrated that elimination of these less important and irrelevant features did not reduce the performance of IDS significantly. Mukkamala *et al.* [12] have demonstrated that use of ensemble of classifiers gave the best accuracy for each category of attack patterns. In designing a classifier, their first step was to carefully construct different connectional models to achieve best generalization performance for the classifiers. Sung and Mukkamala [13] have analyzed data from a large network traffic since it causes a prohibitively high overhead and often becomes a major problem for the IDS.

Chebrolu *et al.* [7] proposed CART-BN approach, where CART performed best for Normal, Probe and U2R and the ensemble approach worked best for R2L and DoS. Meanwhile, A. Abraham *et al.* [14] proved that ensemble of Decision Tree was suitable for Normal, LGP for Probe, DoS and R2L and Fuzzy classifier was good for R2L attacks. A. Abraham *et al.* [15] demonstrated the ability of their proposed Ensemble structure in modeling light-weight distributed IDS.

3. Data Reduction and Feature Selection Using PCA

Principal Components Analysis (PCA) is a predominant linear dimensionality reduction technique, and it has been widely applied on datasets in many different scientific domains [16]. PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional space. The first principal components have the highest contribution to the variance in the original dataset. Therefore, the rest can be disregarded with minimal loss of the information value during the dimension reduction process. Another method is to use their weights and transform data in to a new space with lower dimensions. The transformation works in the following way [17]:

$$X_{M\times N}=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \cdots\cdots & & & \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix}=\left[x_1, x_2, \cdots, x_M\right] \quad (1)$$

Given a set of observations x_1, x_2, $\cdots$, x_M are $N \times 1$ vectors, where each observation is represented by a vector of length N. Thus, the dataset is presented by matrix Equation (1).

The mean value for each column is defined by the expected value. This is explained in Equation (2).

$$\overline{x}=\frac{1}{M}\sum_{i=1}^{M}x_i \quad (2)$$

Once the mean value is subtracted from the data yields expression Equation (3).

$$\varphi_i = x_i - \overline{x} \tag{3}$$

C that is correlation compute from matrix $A = [\varphi_1 \varphi_2 \cdots \varphi_M]$ ($N \times M$ Matrix), Equation (4):

$$A = [\varphi_1 \varphi_2 \cdots \varphi_M] C = \frac{1}{M} \sum_{n=1}^{M} \varphi_n \varphi_n^T = AA^T \tag{4}$$

Sampled $N \times N$ covariance matrix characterizes how data is scattered [18].

The eigenvalues of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$ and the eigenvectors of C: $u_1, u_2, \cdots, u_N$ have to be calculated. Since C is symmetric, $u_1, u_2, \cdots, u_N$ form a basis (*i.e.* any vector x or actually $(x - \overline{x})$ can, can be written as a linear combination of the eigenvectors) Equation (5).

$$x - \overline{x} = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i \tag{5}$$

During the dimensionality reduction, only the terms corresponding to the K largest eigenvalues are mentioned in Equation (6) [19].

$$\hat{x} - \overline{x} = \sum_{i=1}^{K} b_i u_i \quad \text{where} \quad K \ll N \tag{6}$$

The representation of $\hat{x} - \overline{x}$ into the basis $u_1, u_2, \cdots, u_K$ is thus $[b_1 \ b_2 \ \cdots \ b_K]^T$.

The linear transformation $R^N \Rightarrow R^K$ by PCA that performs the dimensionality reduction is presented in Equation (7).

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_K^T \end{bmatrix} (x - \overline{x}) = U^T (x - \overline{x}) \tag{7}$$

The new variables (*i.e.* b_i's) are uncorrelated. The covariance matrix for the b_i's is presented in Equation (8).

$$\mathrm{U}^{\mathrm{T}} C U = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & . & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix} \tag{8}$$

The covariance matrix represents only second order statistics among the vector values.

Let n to be the dimensionality of the data. The covariance matrix is used to calculate $\mathrm{U}^{\mathrm{T}} C U$ that is a diagonal matrix. $\mathrm{U}^{\mathrm{T}} C U$ is sorted and rearranged in the form of $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ so that the data exhibits maximum variance in y_1, the next largest variance in y_2 and so on, with minimum variance in y_n [20,21].

4. K-Nearest Neighbor Algorithm (KNN)

The K-nearest neighbor (KNN) decision rule has been a ubiquitous classification tool with good scalability. Experience has shown that the optimal choice of K is dependent on the data. This makes it difficult to tune the parameters for different applications.

KNN classification algorithm tries to find the K nearest neighbors of x_0 and uses a majority vote to determine the class label of x_0. Without any prior knowledge, the KNN classifier usually applies Euclidean distances as the distance metric [22].

KNN is an example of instance-based learning, in which the training data set is stored, so that, a classification for a new unclassified record may be found simply by comparing it to the most similar records in the training set.

The most common distance function is Euclidean distance, which represents the usual manner in which humans think of distance in the real world (8):

$$d_{\text{Euclidean}}(x, y) = \sqrt{\sum_i (x_i - y_i)^2} \tag{8}$$

where $x = x_1, x_2, \cdots, x_m$, and $y = y_1, y_2, \cdots, y_m$ represent the m attribute values of two records [23,24].

5. Three Way Handshake

The three-way handshake in Transmission Control Protocol (also called the three message handshake) is a method used to establish and tear down network connections. This handshaking technique is referred to as the 3-way handshake or as "SYN-SYN-ACK" (or more accurately SYN, SYN-ACK, ACK). The TCP handshaking mechanism is designed so that two computers attempting to communicate can negotiate the parameters of the network connection before beginning communication. This process is also designed so that both ends can initiate and negotiate separate connections at the same time. Below is a (very) simplified description of the TCP 3-way handshake process (**Figure 2**).

- Source sends a TCP Synchronize packet to destination;
- Destination receives source's SYN;
- Destination sends a Synchronize Acknowledgement packet;

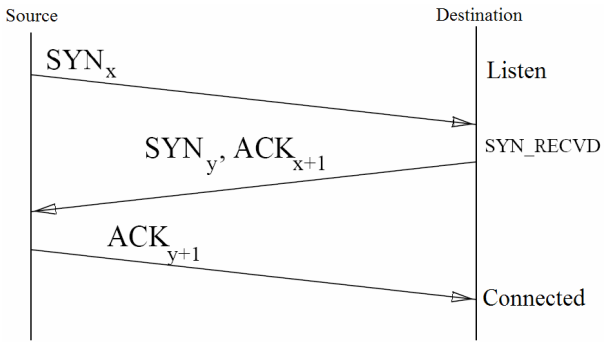

Figure 2. Three-way handshake.

- Source receives destination's SYN-ACK;
- Source sends an Acknowledgement packet;
- Destination receives an Acknowledgement packet;
- TCP connection is established.

Synchronization and Acknowledgement messages are identified by a bit inside the TCP header of the segment. TCP knows whether the network connection is opened, synchronized or established by using the Synchronization and Acknowledgement messages when establishing a network connection.

When the communication between two computers ends, another 3-way communication is performed to tear down the TCP connection. This setup and teardown of a TCP connection is part of the reason why TCP qualifies to be a reliable protocol [25].

6. The Dataset Used in This Work

The DARPA'98 dataset was used for the training dataset in the reported work. The dataset provides around 4 gigabytes of compressed TCP dump data [26] for 7 weeks of the network traffic [27]. This dataset can be processed into about 5 millions of connection records each about 100 bytes in size. Dataset contains payload of the packets transmitted between hosts inside and outside a simulated military base. BSM[1] audit data from one UNIX Solaris host for some network sessions are also provided. DARPA 1998 TCP dump dataset [28] was preprocessed and labeled using two class labels, e.g., normal and attack.

7. Pre-Processing

In this work 32 basic features are extracted from TCP/IP header protocols. These features are derived from TCP, IP, UDP and ICMP packet headers without inspecting the payload. The possible candidates for this feature category includes timestamp, source port, source IP, destination port, destination IP, flag, to name a few. In another dataset 116 derived features are selected from TCP dump network traffic dataset [28]. This dataset is intended to provide a wide variety of features characterizing flows. This includes simple statistics about packet length and inter-packet timings, and information derived from the transport protocol (TCP) such as SYN and ACK counts. This information is extracted using all the packets transmitted in both directions as well as on each direction individually (server → client and client → server).

Many packet statistics are derived directly by counting packets, and packet header-sizes. A significant number of features (such as estimates of round-trip time, size of TCP segments, and the total number of retransmissions) are derived from the TCP headers. TCP trace [29] was used for this information.

Each object within dataset represents a single flow of TCP packets between client and server.

All of the features that are extracted in this work are displayed in **Appendix 1**, **Table A.1**. Wire-shark, Edit-cap and TCP trace softwares are used to analyze and minimize TCP dump files and extract features [30,31].

The dataset contains 13 different types of attacks that are broadly categorized into five groups such as DoS, U2R, R2L, Probing and anomalous behavior. Goal is categorize different intrusion methods into a number of categories. This approach aims to summarize the intrusion method into a few similar approaches. Following the proposed approach, system will be able to deal with variations of the different attacks within each category. Considering the DARPA'98 dataset, there are five main categories of attacks proposed in this paper. The proposed attack categories are listed and described in the following sections.

7.1. Denial of Service (DoS) Attacks

Denial of service attacks consume a large amount of resources thus preventing legitimate users from receiving service with some minimum performance or they may prevent a computer from complying with a legitimate requests by consuming its resources [32,33]. Apache2, Back, Land, Mail bomb, SYN Flood, Ping of death, Process table, Smurf, Teardrop, Udpstorm and Neptune attacks are some examples of the Dos attack. In this work Syn flood attack is used for the experiments. Therefore, Syn flood scenario will be explained in this section: Syn flood is a DoS attack in which every TCP/IP implementtation is vulnerable to it in some degree. Each half-open TCP connection made to a machine will cause the "tcpd" server to add a record to the data structure that stores information describing all pending connections (**Figure 3**). This data structure has a size limit and it may overflow by intentionally creating too many partially-open connections. The half-open connections data structure on the victim server system will eventually fill up. Once the data structure is full, unless the table is emptied, the system will not be able to accept any new incoming con-

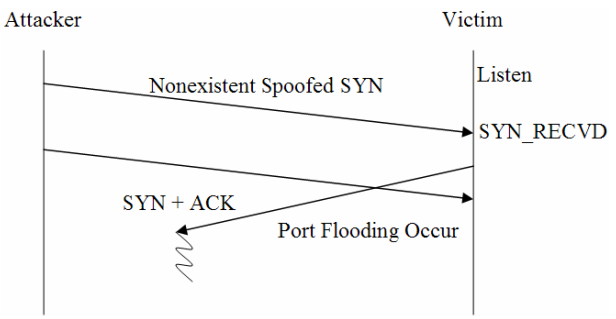

Figure 3. Attacking a victim machine with half-open connections.

[1]Basic Security Monitoring (BSM).

nections [34].

Normally, there is a time-out associated with a pending connection, so that, half-open connections will eventually expire and the victim server system will recover. However, the attacker system can simply continue sending IP-spoofed packets requesting new connections faster than the rate victim system can drop the pending connections. Christopher [35] believes that "Typical SYN flooding attacks can vary several parameters: the number of SYN packets per source address sent in a batch, the delay between successive batches, and the mode of source address allocation".

7.2. User to Root Attacks (U2R)

In this attack, an attacker starts with accessing a normal user account on the system and will end in gaining root access on that system. Regular programming mistakes and environment assumption give an attacker opportunity to exploit the vulnerabilities that may lead to a root access. An example of this type of attacks include buffer overflow, Eject, Ffbconfig, Fdformat, Loadmodule, Perl, Ps, Xterm, perlmagic and ffb attacks [36].

7.3. Remote to User Attacks (R2L)

In this attack, an attacker sends packets to a machine over a network and exploits the machine's vulnerability to gain local access as a user illegally. There are different types of R2U attacks; the most common attack in this class is carried out using social engineering. Examples for these types of attacks are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock, Xsnoop, guessing password and Dict attacks [36].

7.4. Probing Attacks

Probing is a class of attacks where an attacker scans a network to gather information for the purpose of finding known vulnerabilities. An attacker with a map of machines and services that are available on a network can manipulate the information and look for exploits. There are different types of probing, some of them abuse the computer's legitimate features; others use social engineering techniques. This class of attacks is the most common because it requires very little technical expertise. Examples are Ipsweep, Mscan, Nmap, Saint, Satan, pingsweep and Portsweep attacks [6].

7.5. Undetermined Anomalous Behavior

There are anomalous user behaviors, such as "a manager becomes (*i.e*. behaves like) a system administrator". For example, when your computer was automatically blacklisted (blocked) by the network due to the number of abnormal activities originating from your connection, it is possible that your computer is infected with a worm and/or virus.

8. Misuse Detection

Training data from the DARPA dataset includes "list files" that identify the timestamp, source host and port, destination host and port, and the name of each attack [37-40]. This information is used to select intrusion data for the purpose of pattern mining and feature construction, and to label each connection record with "normal" or "attack" label types. The final labeled training data is used for training the classifiers. Due to the large volume of audit data, connection records are stored in several data files. **Table 1** shows 43418 basic feature samples and 20095 derived feature samples that include records from both attack and normal state categories that are selected for the analysis. These data are extracted from the fifth day of the sixth week. Sequences of normal connection records are randomly extracted to create the normal dataset.

Dictionary table is used to convert text data into numeric data.

9. Experiments

Experiments were aimed on generating a categorized attacked or normal state dataset. In the experiments for basic features, 9459 normal connections and 33,959 attacks are included in the categorized attack and were randomly selected to create a dataset. As for the derived features, 10,413 normal connections and 9682 are included in the categorized attack and were randomly selected to create another dataset. With these dataset that included derived features, all experiments repeated again and selected some derived feature in attacks categorized.

Classes of the relevant features with their associated information value are reported in **Tables 2** and **3**. In these tables, all attack categories are compared versus the normal state. As it is reported in this paper, some different features were selected from attacks categories and

Table 1. Number of records that are used for the calculations in different categories.

Category	Number of basic Records	Number of derived records
DoS	19,440	8789
U2R	513	16
R2L	3798	867
Prob	10,137	10
Anomaly	71	0
Normal	9459	10,413
SUM	43,418	20,095

Table 2. List of the most effective basic features for detecting a list of attacks.

Class name	Relevant features in descending order	Total information value
DoS	28,19,5,1,16	99.75%
U2R	12,13,25,28,5	98.13%
R2L	27,25	97.69%
Probing	29,26,25,28,12,13,5,27,1,10	98.01%
Non-deterministic Anomaly	26,28,25,2,3,10,19	99.29%
Normal	27,25	98.84%

Table 3. List of the most effective derived features for detecting a class of attacks.

Class name	Relevant features in descending order	Total information value
DoS	2	99.36%
U2R	79,97,101,10,86,59,47	94.5%
R2L	36,3,77	88.5%
Probing	2,3,35,37,38,61,62,87,89,90,103,104, 102,86,47,10.83	96.24%
Normal	105,99,23,107,103,89	99.22%

normal state. A comparison between the feature importance in different attack categories and the normal state is presented in **Figures 4** and **5**. The Scree graph for the calculated PCA coefficients is depicted in **Figures 6** and **7**.

10. Experimental Results

Each attack has a different consequence and effect on computer network features. Aforementioned features are used to compare each session against a normal or a known attack behavior. **Table 2** for basic and **Table 3** for derived features show relevant features in descending order for different attack categories. As reported in **Table 2**, one single feature (number 27) in normal behavior have 98.22% information value, this is maximum information.

Value in the normal dataset. Once the component number 25 is included, their total information value will rise to 98.84% of the total information value. Therefore, it can be said that the component number 25 does not have a significant effect in detecting the normal state. Comparing information value of the component number 25 versus threshold value for the normal state and R2L attack, normal state and R2L attack can be separated. In the derived features, six features *i.e.* features: 105, 99, 23, 107, 103 and 89 have 99.22% information value for the normal behavior.

As the three-way handshaking was explained in Section 5, intruder may use Syn Flag for the intrusion. The experimental result shows that component number 28 *i.e.* Syn Flag (**Appendix 1**, **Table A.1**) have the highest

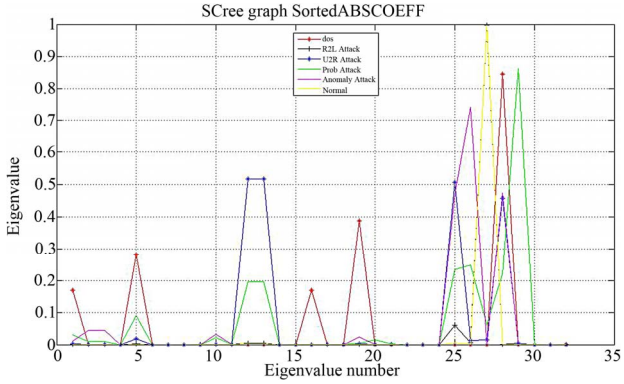

Figure 4. A comparison between the information value of different features in different states of the network operation (basic features).

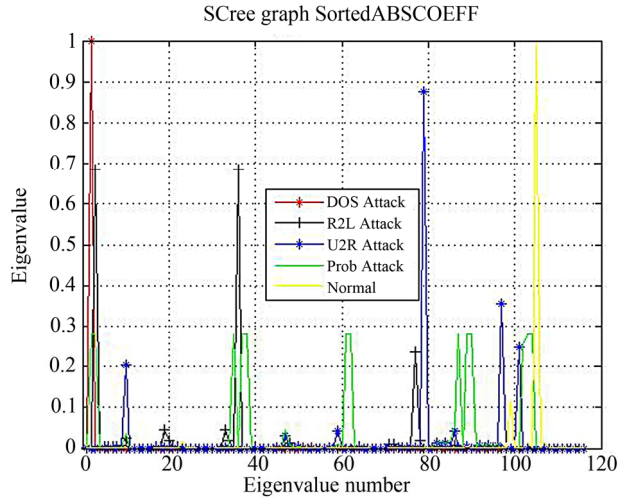

Figure 5. A comparison between the information value of different features in different states of the network operation (derived features).

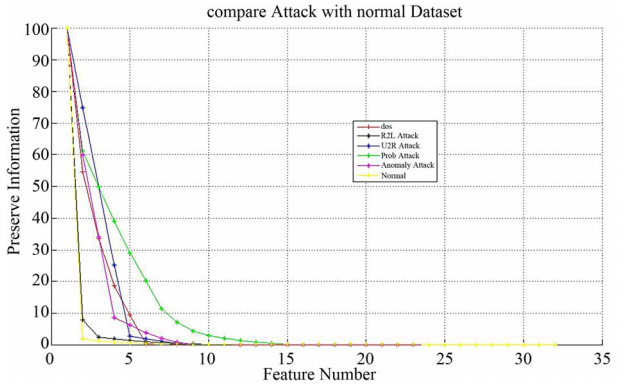

Figure 6. Comparison between Scree graphs for the different calculated PCA coefficients (basic features).

information value for the detection of a DoS attack. Once DoS attack scenarios are compared against the effective features presented in **Table 2**, a relation between the behaviors of their parameters can be extracted.

In TCP scan attack, hackers use TCP scans to identify active devices, TCP port status and their TCP-based application-layer protocols. In TCP FIN scan, that is a kind of TCP scan attack, hackers scan the network to identify TCP port numbers that are listening. The TCP packets used in this scan have only their TCP FIN flag set. Results from the experiments in **Table 2**, for probing attacks, show that the 29th component in **Table A.1** *i.e.* Fin flag has the highest information value. Hence, it is the most important component in the probing scenario attack and for the detection purpose. Comparing results of this experiment with TCP FIN scan scenario, intrusion attempt by probing attack can be detected. In **Table 2**, result of the probing attack scenario shows that the first four components are TCP flags with 70.97% of information value.

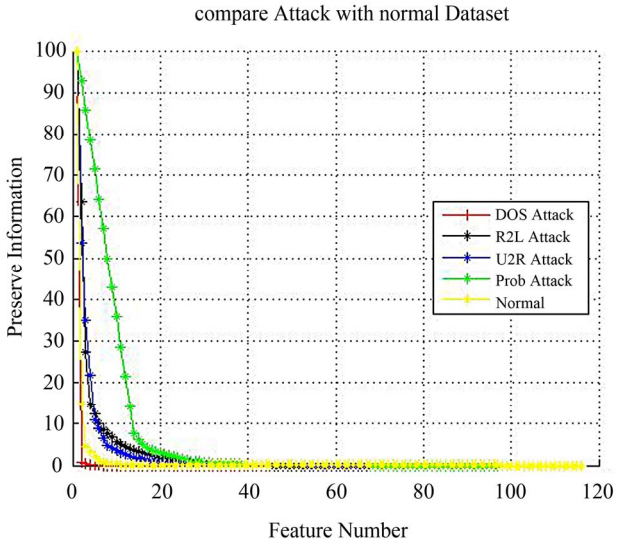

Figure 7. Comparison between Scree graphs for the different calculated PCA coefficients (derived features).

KNN classification method was implemented to show the performance of the proposed measures and to prove that feature reduction will speed up the training and the test processes for the attack identification system considerably. **Table 4** shows the confusion matrix for applying the KNN classification method. In **Table 5**, the classification time for the experiments using all the features are compared with when only effective features are used. True positive and false positive for six classes reported. Once the detection time for the two different feature sets are compared, the result shows that using effective features, the detection time is reduced without any decline in the detection accuracy. Hence, detection time can be reduced using effective features extracted by means of the PCA. In a different experiment, all the attacks in **Table 6** are categorized in an attack class and normal connections are categorized as the second category and the KNN classification method was applied. Process time in this experiment decreased as well, while the accuracy showed a small change.

Table 4. Confusion matrix resulted from implementing KNN classification.

	NOR	DoS	R2L	U2R	PROB	ANOM
NOR	8429	21	8	27	22	6
DoS	0	17,510	0	0	0	0
R2L	10	11	438	2	0	0
U2R	11	0	53	3342	12	0
PROB	40	38	0	5	9040	0
ANOM	1	0	0	2	5	55

Table 5. Comparing classification time for all the features versus when only effective features are used.

Class name	Class 1		Class 2		Class 3		Class 4		Class 5		Class 6		Process time (second)
Record type	Normal		DoS		U2R		R2L		Prob		Anomaly		
Number of record	9459		19,440		513		3798		10,137		71		
Result	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	
KNN classification with all feature	99.01	0.98	100	0	95.01	4.6	97.77	2.2	99.09	0.91	87.30	13.11	200.98
KNN classification with effective feature	98.35	0.84	99.78	0.2	92.03	6.9	94.98	9.0	97.04	0.44	85.2	14.21	135.98

Table 6. A comparison between classification time needed when all features are used or once only the effective features are used.

Class name	Class 1		Class 2		Process time (second)
Record type	Normal		Attack		
Number of record	9459		33,959		
Result	TP	FP	TP	FP	
KNN classification with all feature	99.01	0.99	99.82	0.17	185.32
KNN classification with effective feature	94.05	4.3	99.36	0.58	125.22

11. Conclusions

A feature selection method based on Principal Component Analysis (PCA) for CBID is proposed and implemented to provide results with a similar accuracy but with a smaller set of features. The proposed approach improved the detection speed. Feature selection reduced the total number of features in the dataset (32 basic features and 116 derived features). Due to the smaller search space, this reduction means that less data is needed for training the classifier. Paper reports a new CBID approach that can produce better and more accurate results in identifying the category of the attacks instead of the precise type of the attack. This result also indicates that there are analytical solutions for the feature selection that are not based on the trial and error. The possibility and feasibility of detecting intrusions based on characterization of different types of attacks such as DoS, probes, U2R and R2L attacks is an important goal in the reported work. Results of this investigation seem to be promising.

Results indicate that normal state of the network and category of the attacks can be identified using a small number of a carefully selected network features. On the other hand, it is proven that certain features have no contribution to intrusion detection. Experimental results show that dimension reduction and identification of effective network features for category-based selection can reduce the process time in an intrusion detection system while maintaining the detection accuracy within an acceptable range.

12. Future Work

Plan for the future work is to use different classification methods to detect intrusions. Using the results derived from the intrusion detection and comparing it versus both the full and the reduced feature sets, one can analyze the differences in their accuracy and speed. Also merging KDD Cup 99 features with 116 newly derived features to generate one single dataset and repeat all the experiment for the new dataset and to compare the result with the result reported in this paper.

REFERENCES

[1] A.-B. Amparo, S.-M. Noelia, M. C.-F. Félix, A. S.-R. Juan and P.-S. Beatriz, "Classification of Computer Intrusions Using Functional Networks—A Comparative Study," *Proceedings of European Symposium on Artificial Neural Networks* (*ESANN*), Bruges, 25-27 April 2007, pp. 579- 584.

[2] R. Heady, G. Luger, A. Maccabe and M. Servilla, "The Architecture of a Network Level Intrusion Detection System," Technical Report, University of New Mexico, Albuquerque, 1990.

[3] K. Ilgun, R. A. Kemmerer and P. A. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection Approach," *IEEE Transaction on Software Engineering*, Vol. 21, No. 3, 1995, pp. 181-199. doi:10.1109/32.372146

[4] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, Vol. 3, 2003, pp. 1157-1182.

[5] T. S. Chou, K. K. Yen and J. Luo, "Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms," *International Journal of Computational Intelligence*, Vol. 4, No. 3, 2008, pp. 196-208.

[6] G. Zargar and P. Kabiri, "Identification of Effective Network Feature for Probing Attack Detection," *Proceedings of* 1*st International Conference on Network Digital Technologies*, July 2009, pp. 405-410. doi:10.1109/NDT.2009.5272124

[7] S. Chebrolu, A. Abraham and J. Thomas, "Feature Deduction and Ensemble Design of Intrusion Detection Systems," *Computers and Security*, *Elsevier Science*, Vol. 24, No. 4, 2005, pp. 295-307.

[8] A. H. Sung and S. Mukkamala, "Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks," *Proceedings of International Symposium on Applications and the Internet* (*SAINT*), 2003, pp. 209-216.

[9] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining applications," *Proceedings of Acm-sigmod International Conference on Management of Data*, Seattle, 1998, pp. 94-105.

[10] M. F. Abdollah, A. H. Yaacob, S. Sahib, I. Mohamad and M. F. Iskandar, "Revealing the Influence of Feature Selection for Fast Attack Detection," *International Journal of Computer Science and Network Security*, Vol. 8, No. 8, 2008, pp. 107-115.

[11] B. Chakraborty, "Feature Subset Selection by Neuro-Rough Hybridization," Lecture Notes in Computer Science (LNCS), Springer, Hiedelberg, 2005.

[12] S. Mukkamala, A. H. Sung and A. Abraham, "Modeling Intrusion Detection Systems Using Linear Genetic Programming Approach," Lecture Notes in Computer Science (LNCS), Springer, Hiedelberg, 2004.

[13] A. H. Sung, and S. Mukkamala, "The Feature Selection and Intrusion Detection Problems," Lecture Notes in Computer Science (LNCS), Springer, Hieldelberg, 2004.

[14] A. Abraham and R. Jain, "Soft Computing Models for Network Intrusion Detection Systems," Springer, Hiedelberg, 2004.

[15] A. Abraham, C. Grosan and C. M. Vide, "Evolutionary Design of Intrusion Detection Programs," *International Journal of Network Security*, Vol. 4, No. 3, 2007, pp. 328-339.

[16] C. Boutsidis, M. W. Mahoney and P. Drineas, "Unsupervised Feature Selection for Principal Components Analysis," *Proceedings of the* 14*th ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, Las Vegas, 2008, pp. 61-69. doi:10.1145/1401890.1401903

[17] W. Wang and R. Battiti, "Identifying Intrusions in Com-

puter Networks Based on Principal Component Analysis," 2009. http://eprints.biblio.unitn.it/archive/00000917/

[18] R. D. Jain and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, 2000, pp. 4-37. doi:10.1109/34.824819

[19] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, 1991, pp. 71-86. doi:10.1162/jocn.1991.3.1.71

[20] K. Ohba and K. Ikeuchi, "Detectability, Uniqueness, and Reliability of Eigen Windows for Stable Verification of Partially Occluded Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 9, 1997, pp. 1043-1048. doi:10.1109/34.615453

[21] H. Murase and S. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance," *International Journal of Computer Vision*, Vol. 14, 1995, pp. 5-24.

[22] Y. Song, J. Huang, D. Zhou, H. Y. Zha and C. L. Giles, "IKNN: Informative K-Nearest Neighbor Classification," Springer Verlag, Hieldelberg, 2007.

[23] D. Hand, H. Mannila and P. Smyth, "Principles of Data Mining," MIT Press, Cambridge, 2001.

[24] D. T. Larose, "Discovering Knowledge in Data: An Introduction to Data Mining," John Wiley and Sons Ltd., Chichester, 2005.

[25] 2009. http://support.microsoft.com/kb/172983

[26] 2009. http://www.Tcpdump.org MIT Lincoln Laboratory, 2009. http://www.ll.mit.edu/IST/ideval/

[27] MIT Lincoln Laboratory, Information Systems Technology Group, "The 1998 Intrusion Detection Off-Line Evaluation Plan," 1998. http://www.11.mit.edu/IST/ideval/docs/1998/id98-eval-11.txt

[28] 2009. http://www.wireshark.org

[29] 2009. http://www.Tcptrace.org

[30] 2009. http://www.wireshark.org/docs/man-pages/editcap.html

[31] G. R. Zargar and P. Kabiri, "Category-Based Selection of Effective Parameters for Intrusion Detection," *International Journal of Computer Science and Network Security* (*IJCSNS*), Vol. 9, No. 9, 2009.

[32] A. S. Vasilios and P. Fotini, "Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks," *Proceedings of IEEE Globecom*, 2004, pp. 2050-2054.

[33] A.-B. Amparo, S.-M. Noelia, M. C.-F. Félix, A. S.-R. Juan and P.-S. Beatriz, "Classification of Computer Intrusions Using Functional Networks—A Comparative Study," *Proceedings—European Symposium on Artificial Neural Networks*, Bruges, 2007, pp. 579-584.

[34] A. Hassanzadeh and B. Sadeghian, "Intrusion Detection with Data Correlation Relation Graph," 3*rd International Conference on Availability*, *Reliability and Security* (*ARES* 08), 2008, pp. 982-989. doi:10.1109/ARES.2008.119

[35] L. Christopher, I. Schuba, V. Krsul *et al.*, "Analysis of a Denial of Service Attack on TCP," *Proceedings of the IEEE Symposium on Security and Privacy*, 1997, pp. 208-223.

[36] N. B. Anuar, H. Sallehudin, A. Gani and O. Zakaria, "Identifying False Alarm for Network Intrusion Detection System Using Hybrid Data Mining and Decision Tree," *Malaysian Journal of Computer Science*, Vol. 21, No. 2, 2008, pp. 110-115.

[37] W. Lee, "A Data Mining Framework for Constructing Feature and Model for Intrusion Detection System," Ph.D. Thesis, University of Columbia, New York, 1999.

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

[39] G. R. Zargar and P. Kabiri, "Selection of Effective Network Parameters in Attacks for Intrusion Detection," Lecture Notes in Computer Science (LNCS), Springer, Berlin, 2010.

[40] G. R. Zargar and P. Kabiri, "Identification of Effective Optimal Network Feature Set for Probing Attack Detection Using PCA Method," *International Journal of Web Application* (*IJWA*), Vol. 2, No. 3, 2010.

Table A.1. List of basic features from the TCP/IP protocol with their descriptions in this work.

No.	Feature	Description	
1	Protocol	Type of protocol	
2	Frame_lenght	Length of frame	
3	Capture_lenght	Length of capture	
4	Frame_IS_marked	Frame is marked	
5	Coloring_rule_name	Coloring rule name	
6	Ethernet_type	Type of ethernet protocol	
7	Ver_IP	IP version	
8	Header_lenght_IP	IP header length	
9	Differentiated_S	Differentiated service	
10	IP_Total_Lenght	IP total length	
11	Identification_IP	Identification IP	
12	MF_Flag_IP	More fragment flag	
13	DF_Flag_IP	Don't fragment flag	
14	Fragmentation_offset_IP	Fragmentation offset IP	
15	Time_to_live_IP	Time to live IP	
16	Protocol_no	Protocol number	**Basic feature**
17	Src_port	Source port	
18	Dst_port	Destination port	
19	Stream_index	Stream index number	
20	Sequence_number	Sequence number	
21	Ack_number	Acknowledgment number	
22	Cwr_flag	Cwr flag(status flag of the connection)	
23	Ecn_echo_flag	Ecn echo flag (status flag of the connection)	
24	Urgent_flag	Urgent flag (status flag of the connection)	
25	Ack_flag	Acknowledgment flag (status flag of the connection)	
26	Psh_flag	Push flag (status flag of the connection)	
27	Rst_flag	Reset flag (status flag of the connection)	
28	Syn_flag	Syn flag (status flag of the connection)	
29	Fin_flag	Finish flag (status flag of the connection)	
30	ICMP_Type	Specifies the format of the ICMP message such as: (8 = echo request and 0 = echo reply)	
31	ICMP_code	Further qualifies the ICMP message	
32	ICMP_data	ICMP data	

Appendix 1. Description of the basic and derived features.

No.	Feature	Description	
18	unique_byte_sent_b_a	The number of unique bytes sent the total bytes of data sent excluding retransmitted bytes and any bytes sent doing window probing (server to client)	
19	actual_data_pkts_a_b	The count of all the packets with at least a byte of TCP data payload (client to server)	
20	actual_data_pkts_b_a	The count of all the packets with at least a byte of TCP data payload (server to client)	
21	actual_data_byte_a_b	The total bytes of data seen. Note that this includes bytes from retransmissions/ window probe packets if any (client to server)	
22	actual_data_byte_b_a	The total bytes of data seen Note that this includes bytes from retransmissions/ window probe packets if any (server to client)	
23	rexmt_data_pkts_a_b	The count of all the packets found to be retransmissions (client to server)	Derived feature
24	rexm_data_pkts_b_a	The count of all the packets found to be retransmissions (server to client)	
25	rexmt_data_bytes_a_b	The total bytes of data found in the retransmitted packets (client to server)	
26	rexmt_data_bytes_b_a	The total bytes of data found in the retransmitted packets (server to client)	
27	zwnd_probe_pkts_a_b	The count of all the window probe packets seen (window probe packets are typically sent by a sender when the receiver last advertised a zero receive window to see if the window has opened up now (client to server)	
28	zwnd_probe_pkts_b_a	The count of all the window probe packets seen (window probe packets are typically sent by a sender when the receiver last advertised a zero receive window to see if the window is open now (server to client)	
29	zwnd_probe_byte_a_b	The total bytes of data sent in the window probe packets (client to server)	
30	zwnd_probe_byte_b_a	The total bytes of data sent in the window probe packets (server to client)	

Continued

31	outoforder_pkts_a_b	The count of all the packets that were seen to arrive out of order (client to server)
32	outoforder_pkts_b_a	The count of all the packets that were seen to arrive out of order (server to client)
33	pushed_data_pkts_a_b	The count of all the packets seen with the Push bit set in the TCP header (client to server)
34	pushed_data_pkts_b_a	The count of all the packets seen with the Push bit set in the TCP header (server to client)
35	SYN_pkts_sent_a_b	The count of all the packets seen with the SYN bits set in the TCP header respectively (client to server)
36	FIN_Pkts_sent_a_b	The count of all the packets seen with the FIN bits set in the TCP header respectively (client to server)
37	SYN_Pkts_sent_b_a	The count of all the packets seen with the SYN bits set in the TCP header respectively (server to client)
38	FIN_pkts_sent_b_a	The count of all the packets seen with the FIN bits set in the TCP header respectively (server to client)
39	Urgent_data_pkts_a_b	The total number of packets with the URG bit turned on in the TCP header (client to server)
40	Urgent_data_pkts_b_a	The total number of packets with the URG bit turned on in the TCP header (server to client)
41	Urgent_data_bytes_a_b	The total bytes of Urgent data sent this field is calculated by summing the urgent pointer offset values found in packets having the URG bit set in the TCP header (client to server)
42	Urgent_data_bytes_b_a	The total bytes of Urgent data sent this field is calculated by summing the urgent pointer offset values found in packets having the URG bit set in the TCP header (server to client)
43	mss_requested_a_b	The Maximum Segment Size (MSS) requested as a TCP option in the SYN packet opening the connection (client to server)
44	mss_requested_b_a	The Maximum Segment Size (MSS) requested as a TCP option in the SYN packet opening the connection (server to client)
45	max_segm_size_a_b	The maximum segment size observed during the life time of the connection (client to server)
46	max_segm_size_b_a	The maximum segment size observed during the life time of the connection (server to client)
47	min_segm_size_a_b	The minimum segment size observed during the life time of the connection (client to server)
48	min_segm_size_b_a	The minimum segment size observed during the life time of the connection (server to client)
49	avg_segm_size_a_b	The average segment size observed during the lifetime of the connection calculated as the value reported in the actual data bytes field divided by the actual data packets reported (client to server)
50	avg_segm_size_b_a	The average segment size observed during the lifetime of the connection calculated as the value reported in the actual data bytes field divided by the actual data packets reported (server to client)
51	max_win_adv_a_b	The maximum window advertisement seen if the connection is using window scaling (client to server)
52	max_win_adv_b_a	The maximum window advertisement seen if the connection is using window scaling (server to client)
53	min_win_adv_a_b	The minimum window advertisement seen this is the minimum window scaled advertisement seen if both sides negotiated window scaling (client to server)
54	min_win_adv_b_a	The minimum window advertisement seen. This is the minimum window scaled advertisement seen if both sides negotiated window scaling (server to client)
55	zero_win_adv_a_b	The number of times a zero receive window was advertised (client to server)
56	zero_win_adv_b_a	The number of times a zero receive window was advertised (server to client)
57	avg_win_adv_a_b	The average window advertisement seen, calculated as the sum of all window advertisements divided by the total number of packets seen (client to server)
58	avg_win_adv_b_a	The average window advertisement seen, calculated as the sum of all window advertisements divided by the total number of packets seen (server to client)
59	initial_window_byte_a_b	The total number of byte sent in the initial window the number of bytes seen in the initial flight of data before receiving the first ack packet from the other endpoint (client to server)
60	initial_window_byte_b_a	The total number of bytes sent in the initial window the number of bytes seen in the initial flight of data before receiving the first ack packet from the other endpoint (server to client)
61	initial_window_packets_a_b	The total number of segments (packets) sent in the initial window as explained in above (client to server)
62	initial_window_packets_b_a	The total number of segments (packets) sent in the initial window as explained in above (server to client)
63	ttl_stream_length_a_b	The theoretical stream length, this is calculated as the difference between the sequence numbers of the SYN and FIN packets giving the length of the data stream seen (client to server)
64	ttl_stream_length_b_a	The theoretical stream length, this is calculated as the difference between the sequence numbers of the SYN and FIN packets giving the length of the data stream seen (server to client)

Continued

65	missed_data_a_b	The missed data, calculated as the difference between the ttl stream length and unique bytes sent. If the connection was not complete, this calculation is invalid and an "NA" (Not Available) is printed (client to server)
66	missed_data_b_a	The missed data, calculated as the difference between the ttl stream length and unique bytes sent. If the connection was not complete, this calculation is invalid and an "NA" (Not Available) is printed (server to client)
67	truncated_data_a_b	The truncated data, calculated as the total bytes of data truncated during packet capture. For example, with Tcpdump, the sample option can be set to 64 (with -s option) so that just the headers of the packet (assuming there are no options) are captured, truncating most of the packet data. In an Ethernet with maximum segment size of 1500 bytes, this would add up to total truncated data of 1500, 64 = 1436 bytes for a packet (client to server)
68	truncated_data_b_a	The truncated data, calculated as the total bytes of data truncated during packet capture. For example, with Tcpdump, the sample option can be set to 64 (with -s option) so that just the headers of the packet (assuming there are no options) are captured, truncating most of the packet data. In an Ethernet with maximum segment size of 1500 bytes, this would add up to total truncated data of 1500, 64 = 1436 bytes for a packet (server to client)
69	truncated_packets_a_b	The total number of packets truncated as explained above (client to server)
70	truncated_packets_b_a	The total number of packets truncated as explained above (server to client)
71	data_xmit_time_a_b	Total data transmit time, calculated as the difference between the times of capture of the first and last packets carrying non-zero TCP data payload (client to server)
72	data_xmit_time_b_a	Total data transmit time, calculated as the difference between the times of capture of the first and last packets carrying non-zero TCP data payload (server to client)
73	idletime_max_a_b	Maximum idle time, calculated as the maximum time between consecutive packets seen in the direction (client to server)
74	idletime_max_b_a	Maximum idle time, calculated as the maximum time between consecutive packets seen in the direction (server to client)
75	throughtput_a_b	The average throughput calculated as the unique bytes sent divided by the elapsed time *i.e.*, the value reported in the unique bytes sent field divided by the elapsed time (the time difference between the capture of the first and last packets in the direction) (client to server)
76	throughtput_b_a	The average throughput calculated as the unique bytes sent divided by the elapsed time *i.e.*, the value reported in the unique bytes sent field divided by the elapsed time (the time difference between the capture of the first and last packets in the direction) (server to client)
77	RTT_samples_a_b	The total number of Round-Trip Time (RTT) samples found. TCP trace is pretty smart about choosing only valid RTT samples. An RTT sample is found only if an ack packet is received from the other end point for a previously transmitted packet such that the acknowledgment value is 1 greater than the last sequence number of the packet. Further, it is required that the packet being acknowledged was not retransmitted, and that no packets that came before it in the sequence space were retransmitted after the packet was transmitted. Note: The former condition invalidates RTT samples due to the retransmission ambiguity problem, and the latter condition invalidates RTT samples since it could be the case that the ack packet could be cumulatively acknowledging the retransmitted packet, and not necessarily acking the packet in question (client to server)
78	RTT_samples_b_a	(server to client)
79	RTT_min_a_b	The minimum RTT sample seen (client to server)
80	RTT_min_b_a	The minimum RTT sample seen (server to client)
81	RTT_max_a_b	The maximum RTT sample seen (client to server)
82	RTT_max_b_a	The maximum RTT sample seen (server to client)
83	RTT_avg_a_b	The average value of RTT found, calculated straightforwardly as the sum of all the RTT values found divided by the total number of RTT samples (client to server)
84	RTT_avg_b_a	The average value of RTT found, calculated straightforwardly as the sum of all the RTT values found divided by the total number of RTT samples (server to client)
85	RTT_stdv_a_b	The standard deviation of the RTT samples (client to server)
86	RTT_stdv_b_a	The standard deviation of the RTT samples (server to client)
87	RTT_from_3WHS_a_b	The RTT value calculated from the TCP 3-Way Hand-Shake (connection opening), assuming that the SYN packets of the connection were captured (client to server)
88	RTT_from_3WHS_b_a	The RTT value calculated from the TCP 3-Way Hand-Shake (connection opening), assuming that the SYN packets of the connection were captured (server to client)

Continued

89	RTT_full_sz_smpls_a_b	The total number of full-size RTT samples, calculated from the RTT samples of full-size segments. Full-size segments are defined to be the segments of the largest size seen in the connection (client to server)
90	RTT_full_sz_smpls_b_a	The total number of full-size RTT samples, calculated from the RTT samples of full-size segments. Full-size segments are defined to be the segments of the largest size seen in the connection (server to client)
91	RTT_full_sz_min_a_b	The minimum full-size RTT sample (client to server)
92	RTT_full_sz_min_b_a	The minimum full-size RTT sample (server to client)
93	RTT_full_sz_max_a_b	The maximum full-size RTT sample (client to server)
94	RTT_full_sz_max_b_a	The maximum full-size RTT sample (server to client)
95	RTT_full_sz_avg_a_b	The average full-size RTT sample (client to server)
96	RTT_full_sz_avg_b_a	The average full-size RTT sample (server to client)
97	RTT_full_sz_stdev_a_b	The standard deviation of full-size RTT samples (client to server)
98	RTT_full_sz_stdev_b_a	The standard deviation of full-size RTT samples (server to client)
99	post_loss_acks_a_b	The total number of ack packets received after losses were detected and a retransmission occurred. More precisely, a post-loss ack is found to occur when an ack packet acknowledges a packet sent (acknowledgment value in the ack packet is 1 greater than the packet's last sequence number), and at least one packet occurring before the packet acknowledged, was retransmitted later. In other words, the ack packet is received after we observed a (perceived) loss event and are recovering from it (client to server)
100	post_loss_acks_b_a	The total number of ack packets received after losses were detected and a retransmission occurred. More precisely, a post-loss ack is found to occur when an ack packet acknowledges a packet sent (acknowledgment value in the ack packet is 1 greater than the packet's last sequence number), and at least one packet occurring before the packet acknowledged, was retransmitted later. In other words, the ack packet is received after we observed a (perceived) loss event and are recovering from it (server to client)
101	segs_cum_acked_a_b	The count of the number of segments that were cumulatively acknowledged and not directly acknowledged (client to server)
102	segs_cum_acked_b_a	The count of the number of segments that were cumulatively acknowledged and not directly acknowledged (server to client)
103	duplicate_acks_a_b	The total number of duplicate acknowledgments received (client to server)
104	duplicate_acks_b_a	The total number of duplicate acknowledgments received (server to client)
105	triple_dupacks_a_b	The total number of triple duplicate acknowledgments received (three duplicate acknowledgments acknowledging the same segment), a condition commonly used to trigger the fast-retransmit/fast-recovery phase of TCP (client to server)
106	triple_dupacks_b_a	The total number of triple duplicate acknowledgments received (three duplicate acknowledgments acknowledging the same segment), a condition commonly used to trigger the fast-retransmit/fast-recovery phase of TCP (server to client)
107	max_retrans_a_b	The maximum number of retransmissions seen for any segment during the lifetime of the connection (client to server)
108	max_retrans_b_a	The maximum number of retransmissions seen for any segment during the lifetime of the connection (server to client)
109	min_retr_time_a_b	The minimum time seen between any two (re)transmissions of a segment amongst all the retransmissions seen (client to server)
110	min_retr_time_b_a	The minimum time seen between any two (re)transmissions of a segment amongst all the retransmissions seen (server to client)
111	max_retr_time_a_b	The maximum time seen between any two (re)transmissions of a segment (client to server)
112	max_retr_time_b_a	The maximum time seen between any two (re)transmissions of a segment (server to client)
113	avg_retr_time_a_b	The average time seen between any two (re)transmissions of a segment calculated from all the retransmissions (client to server)
114	avg_retr_time_b_a	The average time seen between any two (re)transmissions of a segment calculated from all the retransmissions (server to client)
115	sdv_retr_time_a_b	The standard deviation of the retransmission time samples obtained from all the retransmissions (client to server)
116	sdv_retr_time_b_a	The standard deviation of the retransmission time samples obtained from all the retransmissions (server to client)

