Intrusion Detection System for High Volume and High Velocity Packet Streams: A Clustering Approach

Dinkar Sitaram, Manish Sharma, Mariyah Zain, Ankita Sastry, and Rishika Todi

***Abstract*—The success of any Intrusion Detection System lies in its ability to quickly adapt to new threats in near real time and further prevent new attacks. This implies extremely efficient machine learning algorithms in the backend, which in turn may use clustering algorithms capable of distinguishing between normal and anomalous network traffic. This work is a first step towards proposing such an IDS, which is built on clustering-based machine learning. The authors evaluate different clustering algorithms using a network packet trace and provide results, which help in evaluating these algorithms. The work-in-progress section of the paper visualizes the IDS which can be used in an environment where the traffic volumes are very high, enterprise boundaries are blurred, and the likelihood of malicious attacks is extremely high.**

***Index Terms*—Clustering, intrusion detection system, security.**

I. Introduction

With the advent of cloud computing and increasingly networked enterprise networks, data sources and sinks have become blurred to the extent that it is almost next to impossible to isolate end points. Hence, it adds to the difficulty of detecting the source of the attack. This is further complicated by the fact that network and cloud infrastructures are shared by disparate organizations and enterprises. This makes it extremely hard for the network and security administrators to analyze the data flow in cases of security attacks. Moreover, the volume of data coming in and going out of a large public cloud is in the order of Petabytes. For example, it is estimated that Amazon EC2 accounts for 1% of all the traffic on the internet [1].

Developing attack prevention and control systems which can handle the volume, velocity, and variety of such data is an interesting challenge. Additionally, with new types of attacks appearing, developing flexible and adaptive security oriented approaches is also a severe challenge. Traditional approaches may not suffice to tackle the attacks prevailing today.

Commercial computer networks administrators rely on special software and hardware systems called Intrusion Detection Systems (IDS). Typically, an IDS monitors traffic on the network and looks for any threats. The goal of an Intrusion Detection System is to automatically scan network activity, differentiate it into ‘attack’ and ‘no attack’, and detect intrusions. Once the attack is detected, the system is expected to alert the administrators so that corrective and preventive actions can be taken.

Historically, signature based IDS’s such as SNORT [2] has been widely used. Such a system monitors packets on the network and compares them against a database of signatures or attributes from known malicious attacks. However, generating the signature of a new attack involves significant processing time and till the new signature is generated and is put to use, the IDS is rendered useless against isolating and protecting from the new threat. Anomaly based IDS’s offer some help in this problem. In network traffic terms, an Anomaly-based IDS captures all the headers of the IP packets entering the network. It then filters out all known and legal traffic, including web traffic to the organization's web server, mail traffic to and from its mail server, outgoing web traffic from company employees and DNS traffic to and from its DNS server. Such systems monitor network traffic and compare it against an established baseline. The baseline identifies a “normality” criteria for that network in terms of bandwidth, protocols, ports, and devices specific to that network. Network and security administrators are alerted in the event when traffic is detected which is anomalous, or significantly different, than the baseline.

In this context, anomaly-based network intrusion detection techniques offer valuable technology to protect target computer systems and networks against malicious activities [3]. However, despite the variety of such methods described in the literature in recent years [4], security tools incorporating anomaly detection functionalities are just starting to appear, and several important problems remain to be solved. For example, anomaly based IDS requires additional hardware spread across the network than is required with other types of IDS’s. Especially larger networks with high bandwidth connections require more hardware and it is therefore necessary to install the anomaly sensors closer to the servers and network that are being monitored. The rationale here is that the amount of data transferred is reduced if the sensors are closer to the application, than if they were located close toor at the network backbone [5].

Data mining methods can help an Intrusion Detection and Prevention system to enhance its performance in various ways such as analysis of stream data, visualization and querying tools, distributed data mining, association, correlation, and discriminative pattern analysis etc.

Clustering algorithms form an important subset of data mining algorithms. These algorithms can analyse network activity and classify it as ‘normal’ or ‘anomalous’. Clustering has been widely referred to as unsupervised learning, which finds ‘natural’ grouping of instances given a set of data. It partitions a given set of data objects into meaningful sub-classes called clusters, such that objects belonging to a cluster are similar to one another, yet dissimilar to objects belonging to other clusters.

Manuscript received June 10, 2013; revised August 15, 2013.

The authors are with the PESIT, Bangalore (e-mail: dinkar.sitaram@gmail.com).

In this work, we make a first attempt towards proposing a clustering based IDS which takes a high volume and velocity network traffic as input (a typical characteristic of a Big Data stream), and employs efficient clustering techniques to train an IDS in a relatively short time using a sub set of the stream. The trained system further detects anomalous behavior based on its learning in the first phase.

The rest of the paper is structured as follows. Section II outlines some of the relevant work done in the area of learning based IDS's. Section III describes our methodology, datasets, and briefly surveys the various clustering algorithms. Section IV presents our experimental results followed by a succinct analysis in Section V. Section VI is one of the key sections of our work where we provide a sneak preview into our work-in-progress. Finally, Section VII presents a summary of our work and some concluding remarks.

II. Related works

A system for intrusion detection based on language models was proposed in [6]. It proposed intrusion detection by extracting language features like n-grams and words from connection payloads, followed by applying unsupervised anomaly detection, i.e. applying clustering algorithms to the extracted data without prior learning phase or the presence of labelled data. One of the key aspects of this work is linear-time computation of similarity measures between language models of connection payloads.

A framework for automatic analysis of malware behaviour using Machine Learning, as proposed in [7], allows automatic identification of novel classes of malware with similar behaviour (clustering) and assigning unknown malware to these discovered classes (classification). Hence the model is capable of processing the behaviour of numerous malware binaries on a daily basis. The incremental analysis significantly reduces the run-time overhead of current analysis methods, while providing accurate discovery and discrimination of novel malware variants.

Intrusion detection can be done in two ways as discussed in [8]: supervised learning and unsupervised learning. Both types of learning were deployed in anomaly detection (unusual activity) and misuse detection (recognize known attack patterns which are signature based). By applying supervised learning for anomaly detection, the resulting algorithms were found to have low efficiency with an exception of SVM and k-nearest neighbors algorithm which proved to have a slightly better efficiency. For misuse detection, the best algorithms found by supervised learning were C4.5 and MLP algorithm. While not one specific algorithm was a winner in the unsupervised learning methodology for both anomaly and misuse detection, Gamma algorithm worked exceptionally well for unknown attacks.

Self learning systems to detect anomalous SIP messages, described in [9]. A self-learning system for anomalous SIP messages is proposed by embedding SIP messages and determining a deviation from a model of normality. The developed model has self-learning capabilities: can automatically retain itself to adapt to moderate changes in network environment and traffic. This was tested on VOIP applications and attacks like Zero Day attacks and worms. The model performed feature extraction: each message was mapped to a feature vector; anomaly detection: feature vectors corresponding to SIP messages are compared against a model of normality; initialization and training: on initial deployment of the system as well as on a periodic basis the learning model is updated using traffic labelled as normal. To prevent external manipulation of the learning process, randomization, sanitization and verification of the model was performed.

Almost all the previous attempts that we have surveyed focus on supervised or unsupervised anomaly detection based on set rules. There is no notion of a prior machine learning phase or labeled data that could be used to train the IDS. Moreover, none of the systems are well suited for anomaly detection in a cloud environment where high volume and high velocity traffic is involved. While our approach uses the fundamentals described in the previous attempts, we focus on detecting anomalies through an efficient clustering algorithm using labeled data which can be further used to train a machine learning system.

III. Methodology and Algorithms

This section describes our approach to demonstrate the effectiveness of clustering algorithms to classify good packets from bad packets. We begin by describing the dataset used for simulating our experiments, and then briefly survey three different clustering algorithms used in our approach, and finally describe the methodology to deploy the clustering algorithms to cluster the sample dataset.

A. Dataset Description

We used the KDD Cup 1999 dataset which consists of 42-tuple instances where each instance represents a raw TCP/IP packet and each attribute/value in an instance is extracted from the respective packet [10]. A few of the attributes present in the dataset are src_bytes, dest_bytes, wrong_fragment, dst_host_count, protocol, service, *et al*. The dataset contains multiple instances and was formed in a simulated military environment. It comprises of 24-types of intrusion attacks which can be categorized under 4 different categories of attacks, viz., Denial of Service- DOS (e.g. Smurf), Remote to Local-R2L (e.g. Password guessing), User to Root-U2R (e.g. Buffer overflow attack), and Probing (e.g. Port scanning).

B. Normalization

We applied z-score normalization (or zero-mean normalization), where the values for an attribute *x*, are normalized based on the mean and standard deviation of *x*. A value *v*, of *x* is normalized to z by computing.

$$z = \frac{x - \mu}{\sigma}$$

where, μ is the mean value and σ is the standard deviation of attribute *x* [11]. Normalization prevents attributes with numerically large values from dominating the clustering.

C. Metric

Determining the appropriate metric is an essential aspect

of any clustering algorithm. For the given problem, we chose the Standard Euclidean Distance [11], since most of the data are continuous numeric values.

D. Clustering

While one aspect of our work is to conceptualize packet classification based on prior machine learning in an environment where the packets arrive in continuous big data like streams, we also want to evaluate various clustering algorithms which are keys to our proposed machine learning techniques. We evaluated the following 3 well-known clustering methods:

Partitioning methods: This method constructs k partitions of the data, where each partition represents a cluster and k<n. The data is divided into k groups, such that each group must contain at least one object. The general criterion of a 'good' partitioning is that objects in the same cluster are 'close' to one another, whereas objects belonging to different clusters are 'far apart'.

Density-based methods: This method has been developed based on the notion of density. The general idea is to continue to grow a given cluster as long as the density in the neighborhood exceeds some threshold.

Model-based methods: This method attempts to optimize the fit between the given data and some mathematical model. They are often based on the assumption that the data are generated by a mixture of underlying probability distributions.

K-Means Clustering, a variant of partitioning methods is a centroid based clustering technique. It is one of the simplest unsupervised learning algorithms that solve the well known clustering problem [11]. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a smart way because different location generates different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early grouping is done. At this point we need to re-calculate k new centroids as centers of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid thereby generating a loop. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function defined as:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2$$

where E is the sum of the squared error, for all objects in the data-set; p is the point in space representing a given object; C_i is the centroid of the cluster, C_i.

To obtain good results in practice, it is common to run the *k*-means algorithm multiple times with different initial cluster centres. The time complexity of *k*-means algorithm is O (nkt), where n is the total number of objects, k is the number of clusters and t is the number of iterations. Therefore the method is relatively scalable and efficient in processing large data-sets.

DBSCAN algorithm, a variant of density-based clustering, works on connected regions with high density [11]. It is done by observing the neighborhood of an object, connecting the objects that have dense neighborhoods to form dense regions as clusters [12]. This algorithm is resistant to noise and can handle clusters of various shapes and sizes. As the DBSCAN algorithm performs clustering based on the density of a point in the object, it has some prerequisites. Before the formation of the clusters actually start, we need to provide two parameters as input along with the object. The parameters are epsilon (eps) and minimum points (min_pts). Epsilon determines the boundary limit for the neighborhood points for the point in consideration, *et al*. points that are at a lesser distance than eps from the point in consideration, are considered as it's set of neighborhood points, that can help it getting added to a cluster provided the set has sufficient number of points . The min_pts specify the minimum threshold for the set of neighborhood points such that the point currently being considered, can become a candidate of a cluster. Density-reachable, core points, border points are a few concepts related to this algorithm. A point p is density-reachable from a point q with respect to eps, min_pts if there is a chain of points $p1, \ldots, pn, p1 = q, pn = p$ such that $pi+1$ is directly density-reachable from pi. An object with at least min_pts objects within a radius 'eps-neighborhood' is a core object. An object that lies on the border of a cluster is a border object [11].

The algorithm is as follows:

1) Arbitrarily select a point (data object) p.
2) Retrieve all points which are density reachable from p wrt eps and min_pts.
3) If p is a core point, a cluster is formed.
4) If p is a border point, no points are density-reachable from o and DBSCAN visits the next point of the database.
5) Repeat steps 2 through 4, until all the instances from the data set have been processed.

If a spatial index is used, the computational complexity of DBSCAN is O (nlogn), where n is the number of database objects. Otherwise, the complexity is O (n2).

Expectation-Maximization algorithm, a variant of model-based clustering, is an iterative refinement algorithm that can be used to find the parameter estimates. EM assigns each object to a cluster according to a weight representing the probability of membership. Hence, new means are computed based on weighted measures. EM starts with an initial estimate or "guess" of the parameters of the mixture model. It iteratively rescores the objects against the mixture density produced by the parameter vector. The rescored objects are then used to update the parameter estimates. Each object is assigned a probability that it would possess a certain set of attribute values given that it was a member of a given cluster [11]. The algorithm is as follows:

Make an initial guess of the parameter vector: This involves randomly selecting k objects to represent the cluster means or centres, as well as making guesses for the additional parameters.

Iteratively refine the parameters (or clusters) based on the following two steps:

1) Expectation step

Assign each object *xi* to cluster *Ck* with the probability

$$P(x_i \in C_k) = p(C_k \mid x_i) = \frac{p(C_k)p(x_i \mid C_k)}{p(x_i)}$$

where $p(xijCk) = N(mk, Ek(xi))$ follows the Gaussian distribution around mean, mk, with expectation, *Ek*.

2) Maximization step

Use the probability estimates from above to re-estimate (or refine) the model parameters. For example,

$$m_k = \frac{1}{n}\sum_{i=1}^{n}\frac{x_i P(x_i \in C_k)}{\sum_j P(x_i \in C_j)}$$

This step is the "maximization" of the likelihood of the distributions given the data.

The E-step takes *O* (*m k d*) operations. Similarly, the M-step takes *O* (*m k d*). The EM algorithm is terminated before a constant number of iterations are used up. The computational complexity of EM algorithm is *O* (*m k d*). Where *m* is the number of sub-clusters, *k* is the number of clusters, *d* is the dimension of data items.

E. Methodology

We used the KDD Cup 1999 dataset, described earlier, for conducting our experiments. Only 10% of the dataset was used which consisted of 6170 instances spanning over a 9-week period. Each instance consists of "label" attribute which is an indication whether the packet is malicious and also lists the type of attack. We used a popular machine learning software WEKA [13] to simulate a scenario where packets are inspected as they flow through an Intrusion Detection System and are further clustered as good or bad packets. WEKA is capable of simulating all three clustering algorithms described above over an arbitrary dataset such as the KDD Cup dataset. We evaluated the 3 clustering techniques mentioned earlier with the same dataset. The next section outlines our results and evaluation.

IV. Evaluation and Results

We first analyzed the dataset and observed the number of normal versus anomalous instances, which in turn indicated benign and malign packets respectively. Table I shows the packet distribution in the dataset.

TABLE I: Packet Distribution in the Dataset

Normal	4498
Anomalous	1672
Total	6170

We evaluated the clustering algoithms using the same dataset and observed the following:

A. K-Means Clustering Algorithm

After setting the number of clusters as 2 and the maximum number of iterations as 500, as in Fig. 1 (where *x*-axis indicates the packet type and y-axis indicates the number of packets) we see that the number of normal packets is 4252 and the number of anomalous packets is 1918. This is very close to the actual number of 4498 and 1672 respectively. No un-clustered packets were found and the time taken to execute the algorithm was 0.575 seconds.

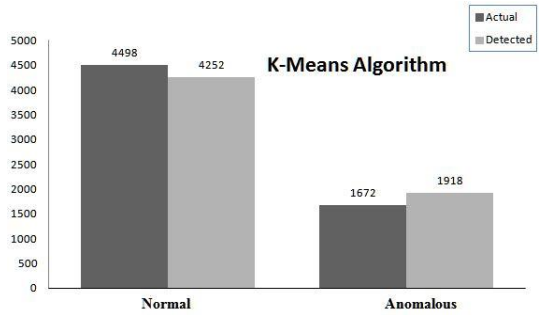

Fig. 1. Actual packets versus detected packets using *K*-means algorithm.

B. Expectation-Maximization Algorithm

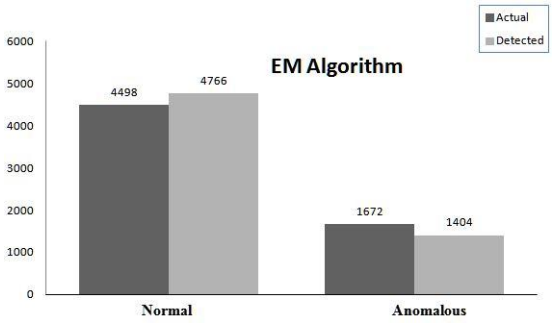

Fig. 2. Actual packets versus detected packets using EM algorithm.

We set the number of clusters as 2 and executed the algorithm for our dataset. As in Fig. 2 (where *x*-axis indicates the packet type and y-axis indicates the number of packets) we found that out of the 6170 instances, 4766 were detected as normal, and 1404 packets were detected as anomalous. There were no un-clustered packets. The time taken to execute the algorithm was 5.26 seconds.

C. DBSCAN Clustering Algorithm

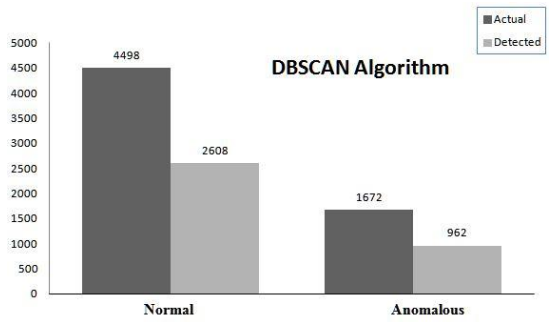

Fig. 3. Actual packets versus detected packets using DBSCAN algorithm.

The values for the parameters required for DBSCAN were taken as follows:

eps- 1.0 min_pts - 800.

The results as in Fig. 3 (where *x*-axis indicates the packet type and y-axis indicates the number of packets) show that out of 6170 instances, 2608 were detected as normal packets and 962 were detected as anomalous packets. The rest of the

packets could not be identified as they were not clustered into any of the two groups. Execution time for the algorithm was 46.6 seconds.

V. Analysis

Our initial experimental results indicate that 2 out of the 3 clustering algorithms produce high fidelity results when compared to the actual packet distribution in the sample dataset. Specifically, the *K*-Means clustering algorithm produced the highest accuracy results amongst the three and was significantly more efficient than the EM and DBSCAN techniques. In fact, its execution time was one order of magnitude faster than the EM algorithm and two orders of magnitude faster than the DBSCAN algorithm. We also evaluated the scalability and efficiency of the *K*-Means algorithm with a large dataset of about 500K instances. The algorithm was able to complete the clustering in about 600-700 seconds while maintaining the same level of accuracy.

VI. Future Work

The purpose of any Intrusion Detection System (IDS) is to be able to detect network attacks in real time (or near real time), isolate the attack, and recover from it. Such a system needs the ability to analyze packets arriving in a network or cloud via fat pipes (OC-192 and OC-768 links) and such data is a clear representation of big data streams in its most raw form (which is hundreds of thousands of TCP/IP packets per second). We envision creating such an IDS where incoming packets are captured, inspected, and analyzed using state-of-the-art Big Data analytics technologies like Hadoop and machine learning algorithms using open source technologies like Mahout.

We are working on one such manifestation of the envisioned IDS using a networking monitoring tool, PacketPig [14], Apache Hadoop [15], and Mahout [16]. PacketPig is a Network Security Monitoring (NSM) Toolset where 'Big Data' is full packet captures. It snoops in on the incoming network packets, through its integration with Snort, p0f and custom java loaders; PacketPig does deep packet inspection (DPI), file extraction, feature extraction, operating system detection, and other deep network analysis. Full packet capture is possible using Apache Hadoop. A standard 100Mbps internet connection can be cheaply logged for months with a 3TB disk. Apache Hadoop is optimized around cheap storage and data locality: putting spindles next to processor cores.

Mahout is used for machine learning in our model and would work on the following lines assuming a network flow (conversation) is defined as 5 tuple defined by source IP, destination IP, source port, destination port, and protocol number:

1) Extract all conversations as 4 tuples or 5 tuples (with protocol number) and extract features out of each conversation to create a vector space.
2) Extract all attacks and join the two data sets so that we now know what conversations are 'attacks' and what are 'not attacks'.
3) Train a model based on this information.
4) Test new traffic or perform cross validation to test the accuracy of your model.

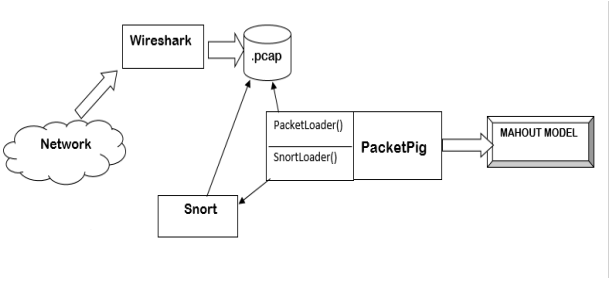

Fig. 4. Schematic of proposed future work.

A schematic of our future work is depicted in Fig. 4.

PacketPig takes pcap files as its input and processes them to extract packet information. This packet information, is given as input to Mahout which deploys the selected clustering algorithm to generate clusters of "good" vs. "bad" packets. We use machine learning techniques in this phase. PacketPig helps in labeling data which in turn is used by machine learning algorithms to train on and test new traffic.

VII. Conclusion

The demonstrated the effectiveness of clustering algorithms with reference to clustering normal versus anomalous network packets. We concluded that *K*-Means clustering algorithm provides the highest level of accuracy with minimum time complexity. Our work is a first step in the direction of building an Intrusion Detection System which is based on the premise that network packets arriving in a public cloud can be treated as big data streams. State-of-the-art data capture and inspection techniques combined with big data processing technologies like Hadoop and through effective use of clustering algorithms deployed in the machine learning phase, a new IDS can be created. Such an IDS will be best suited in a cloud environment which is a sink for heavy network traffic and where network intrusion can be detrimental to the service providers and consumers of the cloud services.

References

[1] How big is Amazon's cloud? [Online]. Available: http://www.deepfield.net/2012/04/how-big-is-amazons-cloud.
[2] Snort: a lightweight network intrusion detection system for windows and unix. [Online]. Available: http://www.snort.org/.
[3] E. Bloedorn, A. Christiansen, W. Hill, C. Skorupa, L. Talbot, and J. Tivel, *Data Mining for Network Intrusion Detection: How to Get Started*, 2001.
[4] F. Gong, "Deciphering detection techniques: part 2 - anomaly-based intrusion detection," March 2003.
[5] Signature-based or anomaly-based intrusion detection- practise and pitfalls. [Online]. Available: http://www.scmagazine.com/signature-based-or-anomaly-based-intrusion-detection-the-practice-and-pitfalls/article/30471/.
[6] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," *Journal in Virology Manuscript*, 2006.
[7] K. Rieck, P. Trinius, C. Williems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, IOS Press, June 2011.
[8] P. Laskov, P. Dussel, C. Schafer, and K. Rieck, "learning intrusion detection: supervised or unsupervised," in *Proc. International Conference on Image Analysis and Processing (ICIAP)*, 2005.
[9] K. Rieck, S. Wahl, P. Laskov, P. Domschitz, and K. R. Muller, "A self-learning system for detection of anomalous sip messages,"

Principles, Systems and Applications of IP Telecommunications; Services and Security for Next Generation Networks, 2008.
[10] A. Olusola, A. Oladele, and D. Abosede, "Analysis of KDD '99 intrusion detection dataset for selection of relevance features," in *Proc. The World Congress on Engineering and Computer Science*, 2010.
[11] J. Han, M. Kamber, and J. Pei, "Data mining: concepts and techniques," *The Morgan Kauffman Series in Data Management Systems*, July 2011.
[12] X.-Y. Li *et al.*, "A new intrusion detection method based on improved DBSCAN," in *Proc. International Conference on Information Engineering (ICIE)*, 2010.
[13] WEKA: a data mining software in java open. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka.
[14] Packet pig: open source big data security analytics on full packet captures. [Online]. Available: https://github.com/packetloop/packetpig.
[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proc. The 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 2010.
[16] Apache mahout: an Apache project to produce implementations of distributed or scalable machine learning algorithms on the Hadoop platform. [Online]. Available: http://mahout.apache.org/.

Dinkar Sitaram is the head of the Center for Cloud Computing and Big Data at PESIT and is a professor in the Department of Computer Science and Engineering at the PES Institute of Technology. His research interests include scheduling and resource allocation policies for cloud computing and hybrid clouds, as well as the usage of cloud computing and big data techniques for audio and video analysis. He is the author of two books - a recent book on Cloud Computing, called "Moving to the Cloud – Developing Apps in the New World of Cloud Computing" published by American Elsevier in December 2011. His earlier book "Multimedia Servers" was published by Morgan Kaufman. He is the author of over 20 patents and 25 publications. Dr. Sitaram received his Ph.D. from the University of Wisconsin-Madison and his B.Tech from IIT Kharagpur. As a researcher at the IBM T.J.Watson Research Center, NY Dr. Sitaram worked on file systems and multimedia servers. He received an IBM Outstanding Innovation Award (an IBM Corporate Award) as well as IBM Research Division Award and several IBM Invention Achievement Awards for his patents and research, and received outstanding paper awards for his work. He also served on the editorial board of the Journal of High-Speed Networking. Subsequently, he returned to India as Director of the Technology Group at Novell Corp., Bangalore. Under his direction, the group developed many innovative products in addition to filing for many patents and standards proposals. Dr. Sitaram received Novell's Employee of the Year award. Subsequently, Dr. Sitaram was CTO at Andiamo Systems India (a storage networking startup), responsible for architecture and technical direction of an advanced storage management solution. Andiamo Systems was acquired by Cisco Systems. Most recently, Dr. Sitaram was CTO at HP Systems Technology and Software Division, the product R&D organization of HP in India. He focused on driving cloud and storage technologies as part of the HP Storage Division. He also worked on file system strategies, and high reliability for the HP Unix server division. He was also responsible for Patents and University Relations, served on HP's global patent committee, and several times on the program committee of HP's internal technical conference.

Manish Sharma was born in Lucknow, India on 14th March 1976. Manish has a Master's degree in computer science from Boston University and a Bachelor's degree in electrical and electronics engineering from Birla Institute of Technology, Ranchi. He has worked in IT organizations like Wipro, Sprint, Oracle, AOL, and Guavus in various roles and has published in international conferences like Globecom and PAM. He is currently working as a Head for Solutions Delivery and Support for Guavus Inc. in India and is also an Assistant Professor at PESIT, Bangalore.

Mariyah Zain was born in Mysore, India in July 1992. She is currently pursuing her final year of Bachelor of Engineering degree in computer science at PES Institute of Technology (west campus), Bangalore, and is expected to receive her degree in the year 2014. Her current research interests lie in the fields of Data Mining, Computer security; Cloud computing and Big Data Analytics. She was awarded second place at an All - India Collegiate Cyber threat competition for her research in the above mentioned fields by a leading global firm in April 2013.

Ankita Sastry was born in Bangalore, India in Feburary 1992. She has done her schooling in Sophia High School, Bangalore and is currently pursuing her final year of Bachelor of Engineering Degree in the field of computer science at PES Institute of Technology (west campus), Bangalore. She has participated in competitions related to cyber security and also came second in an All-India Collegiate Cyber threat competition hosted by a leading global firm, which took place in Hyderabad, India. Her interests lie in the fields of computer security, cloud computing and big data analytics.

Rishika Todi was born in Asansol,West Bengal, India in the year 1991. She has done her schooling in Assembly of God Church School .She is currently pursuing her Bachelor of Engineering Degree in the field of Computer Science at People's Education Society Institute of Technology and is expected to receive her B.E degree in Computer Science in 2014. Her interests include Big Data analytics, network security and machine learning.