Collaborative IDS Framework for Cloud

Dinesh Singh[1], Dhiren Patel[2], Bhavesh Borisaniya[2], and Chirag Modi[3]
(Corresponding author: Dinesh Singh)
Department of Computer Science & Engineering, Indian Institute of Technology, Hyderabad, India[1]
Department of Computer Engineering, National Institute of Technology, Surat, India[2]
Department of Computer Science & Engineering, National Institute of Technology Goa, India[3]
(Email: barehla88@gmail.com)
(Received Apr. 16, 2014; revised and accepted Jan. 16 & Mar. 4, 2015)

Abstract

Cloud computing is used extensively to deliver utility computing over the Internet. Defending network accessible Cloud resources and services from various threats and attacks is of great concern. Intrusion Detection System (IDS) has become popular as an important network security technology to detect cyber-attacks. In this paper, we propose a novel Collaborative IDS (CIDS) Framework for cloud. We use Snort to detect the known stealthy attacks using signature matching. To detect unknown attacks, anomaly detection system (ADS) is built using Decision Tree Classifier and Support Vector Machine (SVM). Alert Correlation and automatic signature generation reduce the impact of Denial of Service (DoS) /Distributed DoS (DDoS) attacks and increase the performance and accuracy of IDS.

Keywords: Anomaly detection, collaborative IDS, cloud security, intrusion detection, signature generation

1 Introduction

Users of a cloud request access from a set of web services that manage a pool of computing resources (i.e., machines, network, storage, operating systems, application development environments, application programs). When granted, a fraction of the resources from the pool they are dedicated to the requesting user until he or she releases it. Cloud computing combines several technologies like distributed computing, grid computing, virtualization, utility computing, network computing etc. Each of the involving technologies has vulnerabilities that cause several security and privacy issues. One of the major security challenges is to defend Cloud network from the attacks like IP spoofing, DNS poisoning, man-in-the-middle attack, port scanning, insider attack, Denial of Service (DoS) attack, and Distributed Denial of Service (DDoS) attack etc. [15].

To deal with such attacks, Intrusion Detection System (IDS) can be used. Intrusion detection is the act of detecting actions that attempt to compromise the Confidentiality, Integrity or Availability of a system/network. Security threats are divided into three categories [20]: (1) breach of confidentiality, (2) failure of authenticity, and (3) unauthorized denial of service.

Based on the protection objective, IDS are classified into three categories: Host-based (HIDS), Network-based (NIDS) and Distributed IDS. Host based IDS collects the internal activities (like system call) of a host and analyse for malicious activities. Network based IDS attempts to discover unauthorized access to a computer network by analyzing network traffic. Distributed IDS collects the events from multiple sources and analyzes collectively for malicious activity. On the basis of detection techniques, IDSs are divided in two categories [7] viz; Signature based and Anomaly based. Signature based IDS detects known attacks through matching signature in pre-stored attack signature base. Signatures are the well formatted patterns found in the attack. Thus they are limited to detecting known attacks. Anomaly based IDS store the behavior of previous events and construct a model to predict the behavior of the incoming events. These systems are able to detect both known as well as an unknown attack, however produce high false alarm and high computational cost. Isolated IDSs are not able to detect coordinated attack such as DDoS attacks. To detect such kind of attacks, we need collaborative IDS. A collaborative IDS framework consists of two main functional units [29]:

1) Detection Unit: A detection unit consists of multiple detection sensors, where each sensor monitors its own sub network or hosts separately and then generates low-level intrusion alerts.

2) Correlation Unit: A correlation unit transforms the low-level intrusion alerts into a high level intrusion report of confirmed attacks. There are three alert correlation approaches:

    a. Centralized approaches [29]: Each participating IDSs has only detection unit, while analysis unit is at the central server.

b. Hierarchical approaches [29]: Each IDS has detection unit. The entire system is organized into a hierarchy of small communication groups. Each group has its correlation unit that is responsible for correlation within the group and its processed data will be sent upward to a node at a higher level in the hierarchy for further analysis.

c. Fully distributed approach [29]: Each participant IDSs has both detection unit and correlation unit and communicates to each other using some protocol like peer-to-peer.

We are using a centralized approach as the importance of communication in cloud computing is vital. In comparison to fully distributed and hierarchical approaches, centralized approach is less scalable, but requires less communication overhead [29].

Shared and distributed resources in the Cloud system make it difficult to develop a security model for detecting intrusion and ensuring the data security and privacy in the Cloud. Because of transparency issue, no Cloud provider allows its customers to implement intrusion detection or security monitoring system extending into the management services layer providing back channel behind virtualized Cloud instances. IDS technology has been tested to be capable of working well in some large scale networks, however, its utilization and deployment in Cloud Computing is still a challenging task [1].

In this paper, we have proposed a Collaborative IDS (CIDS) which keeps the knowledge base up-to-date, produce low communication overhead and able to detect known and unknown attack with fast detection rate.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 describes the theoretical background about classifiers used in our proposed approach. The proposed approach is discussed in Section 4. Section 5 describes the experimental setup, evaluation method and results. Section 6 concludes our research work with references at the end.

2 Related Work

Several IDS have been proposed to-date to detect intrusions in the traditional network and in the Cloud network.

Hwang et al. [8, 9] proposed a cooperative anomaly and intrusion detection system for a distributed network. The signature-based NIDS (Snort) is cascaded with a custom designed ADS. These two subsystems join hands to cover all trafic flow events, initiated by both legitimate and malicious users. Single connection intrusive attacks are detected by NIDS at the packet level by signature matching. Remaining unknown attacks, which cannot be detected by signature-based NIDS, are passed on to the ADS. A signature generator bridges the two sub-systems.

Lo et al. [13] proposed a system to reduce the impact of DOS and DDOS attacks. To provide such ability, IDSs in the cloud computing regions exchange their alerts with each other. In the system, each of IDSs has a cooperative agent used to compute and determine whether to accept the alerts sent from other IDSs or not. By this way, IDSs could avoid the same type of attack happening in future. But this system uses fully distributed alert correlation system which produces high communication overhead.

Modi et al. [16] proposed a framework to reduce the impact of DoS and DDoS which integrates a NIDS in the Cloud infrastructure. They combined Snort and decision tree (DT) classifier to implement their framework. It aims to detect network attacks in Cloud, while maintaining performance and service quality.

Sandar et al. [24] describe a new type of DDoS attack, called Economic Denial of Sustainability (EDoS) in Cloud services and proposed a solution framework for detecting EDoS attack. EDoS attacks are HTTP and XML based DDoS attack. The EDoS protection framework uses firewall and puzzle server to detect EDoS attack. Here, the authors demonstrated EDoS attack in the Amazon EC2 Cloud. However, it is not an adequate solution because it uses only traditional firewalls.

Combining the multiple techniques overcome the limitation of each other. Gaddam et al. [4] proposed a supervised anomaly detection using k-Means clustering and Decision Tree. A method to cascade k-Means clustering and the ID3 decision tree learning methods for classifying anomalous and normal activities in a computer network. First of all using k-Means, the dataset is partition in k clusters. Then the decision tree on each cluster refines the decision boundaries by learning the sub-groups within the cluster. To obtain a final decision on the classification, the decisions of the k-Means and ID3 methods are combined using two rules: (1) the Nearest-neighbor rule and (2) the nearest consensus rule. A similar approach is proposed by Yasami et al. [28] for unsupervised learning. However, the use of a serial combination of k-Means and ID3 increase the learning time. Detection on both Subject to algorithm and rules for final decision has also increased the detection time as well.

3 Theoretical Background

3.1 Snort

Snort [25], is a well-known open source packet sniffer and NIDS. It is configurable and freely available for multiple platforms (i.e. GNU/Linux, Window). The misuse IDS model used in Snort is based on matching of attack signature with pre-stored signatures associated with known attacks like the PoD, port-sweep, DoS-nuke, Tear-drop, and Saint, etc. The detection engine of Snort allows registering, alerting and responding to any known attack. Snort cannot detect unknown or multi-connection attacks [8, 9].

Decision Tree Classifier

Decision tree (DT) classifier [6, 16] is a supervised classification technique. It requires a labelled training dataset to construct a decision tree. As shown in Figure 1, the decision tree is a tree structure, where each non leaf node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label.

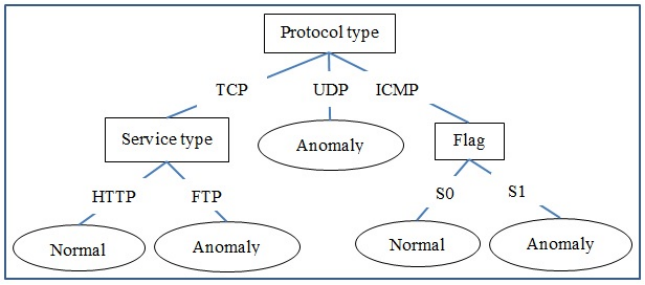

Figure 1: A sample decision tree

To test an unknown network traffic profile tuple (e.g. X), the attribute values of the X are tested against the decision tree. A path is traced from the root to a leaf node; class label of the leaf is the prediction for that tuple X.

For decision tree classifier, no domain knowledge or parameter setting is required, and therefore it is appropriate for exploratory knowledge discovery. It can handle high dimensional data and the representation of acquired knowledge in tree form is intuitive, and generally easy to assimilate by humans [16].In general, decision tree classifiers have good accuracy for categorical data values but in case of continuous data values it suffers from overfitting [22, 27]. However, successful use may depend on the data used for learning.

3.2 Support Vector Machine

Support Vector Machine (SVM) is based on statistical learning theory developed by Vapnik [6, 14]. The SVM approach is very popular for classification and regression problems because of its good generalization capability and its superiority in comparison with other machine learning paradigms. SVM solves the problem of over-fitting and can easily make a generalized model from the least number of samples. But their learning time increases rapidly with an increase in training size. SVMs were originally designed for binary-class classification; hence, it is straightforward to use this paradigm in the present problem for classification between normal and malicious behavior in the patterns of activity in the audit stream. In fact, SVMs [12, 14, 17] have been proposed as a powerful technique for intrusion detection classification. It classifies data by determining a set of support vectors, which are members of the set of training inputs that outline a hyperplane in feature space.

Let us assume $\{(x_1, y_1), ..., (x_n, y_n)\}$ be a training set with $x_i \in R_d$ and $y_i = \{-1, +1\}$ is the corresponding target class. The basic problem for training an SVM can be reformulated as:

$$Maximize : J = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T, x) \quad (1)$$

$$Subject\ to \sum_{i=1}^{n} \alpha_i y_i = 0\ and\ \alpha_i \geq 0, i = 1, 2, ..., n$$

Kernel function is used for computation of dot products between vectors without explicitly mapping to another space. Use of a kernel function [18] addressed the curse of dimensionality and the solution implicitly contains support vectors that provide a description of the significant data for classification. Substituting Kernel $K(x_i^T, x)$ for in Equation (1) produces a new optimization problem:

$$Maximize : J = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i^T, x) \quad (2)$$

$$Subject\ to \sum_{i=1}^{n} \alpha_i y_i = 0\ and\ 0 \leq \alpha_i \leq C, i = 1, 2, ..., n$$

where C is soft margin parameter. Solving it for, gives m support vectors (SV), their respective values of α_i and the value of bias b. These SVs gives a decision function of the form

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i K(x_i^T, x) + b, \quad (3)$$

where α_i are Lagrange multipliers, x is the test tuple and $f(x) = f(-1, +1)$ is its prediction.

4 Proposed CIDS Framework

As shown in Figure 2, we integrate NIDS module in each cloud cluster to detect network attacks. Correlation Unit (CU) is placed in any one cluster. NIDS detects the intrusions within a cluster and Correlation Unit provides collaboration between all cluster NIDSs. Bully [5] election algorithm is used to elect one best cluster for placement of CU on the basis of workload.

4.1 NIDS Architecture

As shown in Figure 3, we use Snort and an Anomaly Detection System (ADS) built using Decision Tree classifier and SVM classifier techniques. Snort is used to detect known attacks, whereas ADS predicts that the given event is malicious or not, by observing previously stored network events.

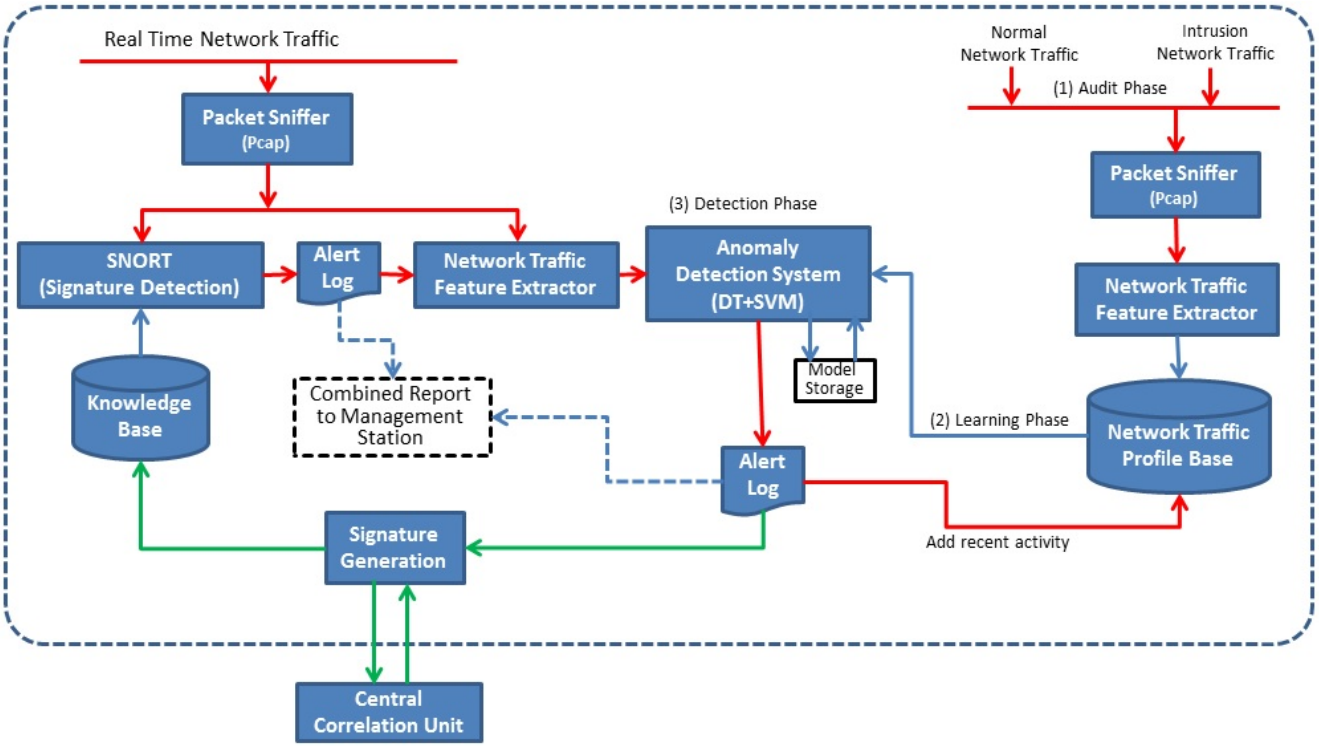

Figure 3: NIDS architecture

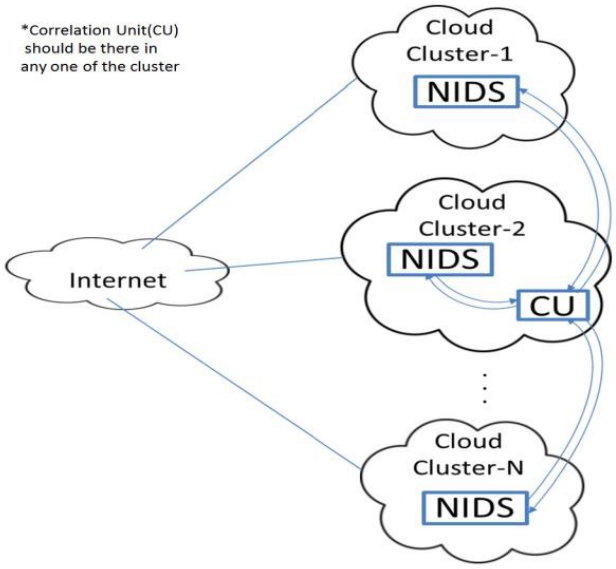

Figure 2: Proposed collaborative IDS framework in cloud

1) Audit Phase. During the audit phase, various (normal and intrusion) network traffic profiles are generated and stored. First we capture the normal traffic and generate network traffic profiles and give them class label as Normal. To generate malicious traffic, we perform various attacks and again capture the traffic and generate network traffic profiles and give them class label as Intrusion and store into the network traffic profile base. Network profile generation process is explained in Section 4.3.

2) Learning Phase: In this phase, a model for anomaly detection system is constructed from the network traffic profile base. The learning process of Anomaly Detection is shown in section 4.2.

3) Detection Phase: During the detection phase, we capture the real time traffic and generate network traffic profiles on the y and pass these profiles as input to the ADS. ADS generates the alert, if it found any correlation of the input profile with malicious profiles.

Incoming network traffic will pass through Snort; here known attacks are identified through signature matching. The remaining attacks are detected by ADS. An alert entry is made in the log, if an unknown attack is detected. If the frequency of an attack detected by ADS is crossing a frequency threshold T_f, then we go for generating a Snort based signature for those connections. This increases the performance of NIDS as Snort is able

to detect these frequent attacks in a short time. Once the signature is generated, we update local knowledge base as well as send this signature to a central correlation unit. The central correlation unit receives the signature sent by all the NIDSs in the Cloud network and make a decision on the bases of how much part of total NIDSs send the similar signature.

$$S > S_T \text{ where } S = \frac{No\ of\ IDS\ Support\ Same\ Signature}{Total\ No\ of\ IDS\ in\ the\ System}$$

$$and\ S_T = Threshold.$$

Value of S_T will be set by admin (as 0.5 for majority decision). If $S > S_T$ for an attack signature then correlation unit multicasts this signature to all the IDSs. They receive this signature and update their knowledge base.

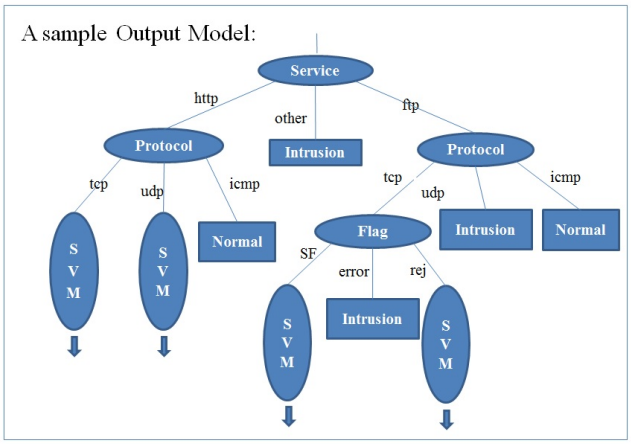

Figure 4: A sample model for anomaly detection

4.2 Proposed Anomaly Detection System

We split the training dataset using decision tree and build the SVM model on each subset. First, we call decision tree algorithm for attributes having categorical data values. We select a best attribute on the basis of maximum information gain and make the root node of the tree to use this attribute. The branches of this node are the distinct values of the selected attributes. These branches end on some other node. Then we split the entire data set into subsets with respect to each distinct value of selected attribute. We call the decision tree algorithm for each sub dataset recursively. If at some place, all profiles belong to a same class label then the leaf node with that class label is created, if not, then another attribute of categorical data values is selected to create an internal node like root node. If at any stage, no attribute with categorical data values remaining or the information gain of best attribute chosen is less than the threshold then a model is created using SVM for the continuous values. The output looks like as shown in Figure 4. The learning process is shown in Figure 5.

4.2.1 Learning Algorithm

Algorithm 1 Learning algorithm
$\mathbf{D}$ = *Set of Network Traffic Profiles used for training.*
$\mathbf{C}$ = *Set of Class Labels i.e. Intrusion, Normal.*
$\mathbf{A}$ = *Set of Attribute used to represent Network Connection Profiles.*
We divide the attributes into two subsets,
$\mathbf{A_S}$ = *Set of Symbolic (Categorical) value Attributes (e.g. Protocol, Service, flag etc.).*
$\mathbf{A_N}$ = *Set of Numeric (Continuous) value Attributes (e.g. Srcbyte, Dstbyte, count etc.).*
$\mathbf{T_{InfoGain}}$ = *Minimum Threshold for InfoGain*
$\mathbf{H}$ = *Hyperplane*

$$\mathbf{InfoGain(D)} = E(D) - \sum_{i=0}^{v} \frac{|D_i|}{|D|} E(D_i)$$

$$where\ \mathbf{E(D)} = -\sum_{i=0}^{m} p_i log_2(p_i)$$

E is the entropy and is the probability of appearance of Class label.
DecisionTree(D, A_S, A_N)

1: Begin
2: **if** (All Samples in $D \in C_i$) **then**
3: Create Leaf Node with Class Label C_i;
4: **end if**
5: **if** $(A_S = \phi)$ **then**
6: $H \leftarrow SVM(D, A_N)$; //construct the SVM model
7: Create Leaf Node with H;
8: **end if**
9: $A_{S-best} \leftarrow getBestAttribute(D, A_N)$;
10: **if** $A_{S-best}.InfoGain \leq T_{InfoGain}$ **then**
11: $H \leftarrow SVM(D, A_N)$;
12: Create Leaf Node with H;
13: **end if**
14: $Root \leftarrow createNode(A_{S-best})$;
15: $A_S \in A_S - A_{S-best}$;
16: **for** each value $V_i \in Domain(A_{S-best})$ **do**
17: $D_i \leftarrow D\ where\ (A_{S-best} = V_i)$;
18: $ChildTree \leftarrow$ DecisionTree(D_i, A_S, A_N);
19: Root.Child[i] $\leftarrow$ ChildTree;
20: Return *Root*
21: **end for**
22: End

4.2.2 Testing

To test an unknown profile on ADS, we trace the tree from root to leaf; if leaf node is a class label then this is the prediction. If a leaf node is an SVM model then the prediction is given by this SVM model.

4.3 Network Traffic Profile Generation

A packet sniffer (libpcap) is used to capture network packet frames from the data link layer and to assemble

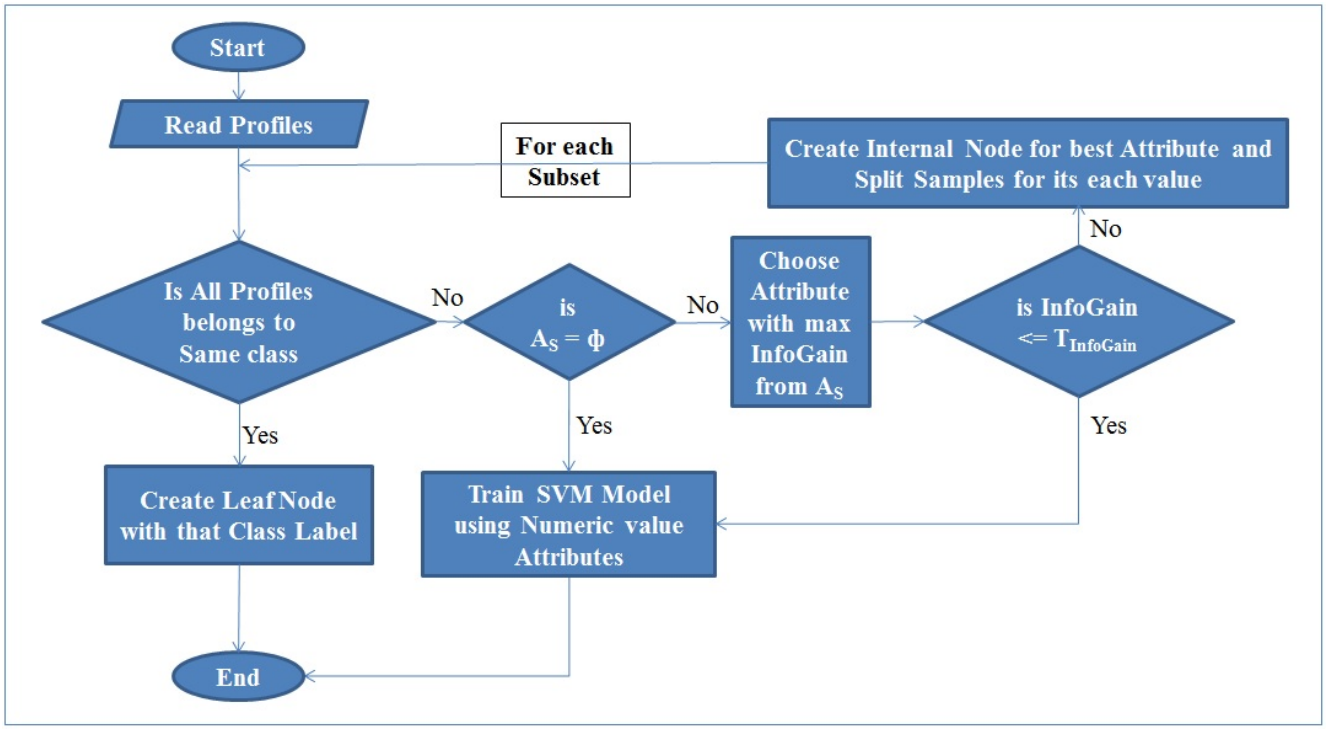

Figure 5: Flow chart of learning process of ADS

them as raw packet. The packets are collected for a complete connection. A connection is a sequence of packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address under some well-defined protocol [11]. For generating a network profile, the network traffic feature extractor extracts the network features viz; basic, traffic and content (as in KDD'99 dataset) from the raw packets [11].

1) Basic Features: It involves all the attributes that are extracted from a TCP/IP connection, e.g., protocol, service, size of traffic flow etc.

2) Traffic based Features: These features are computed within time frame, and divided into two groups viz; same host features and same service features. Same host features involve the connections having same destination host within given time frame (E.g. 2 seconds) and statistics related to protocol, service, flag error etc. Same service features include the connections having same services within given time frame to calculate traffic related statistics.

3) Content based Features: In this category, data portions of the packets are examined. It involves only a single connection. To detect attacks (E.g. Remote to local and User to root) that are embedded in the data portions of the packets, suspicious behavior in the data portion is looked, e.g., number of failed login attempts, number of root access.

A Connection is identified as ($SrcIP$: $SrcPort$ $\rightarrow$ $DstIP$: $DstPort$ $Protocol$). As as soon as a new connection starts, we make an entry into Connection cache and capture all packets sent during communication. When the connection terminates then we extract basic features from header part, content features from payload and traffic statistics by comparing this connection with the previously established connection (during last t seconds). Where, t is the size of the sliding window.

4.4 Signature Generation

As shown earlier in Figure 3, signature generation is an independent process running side by side. For frequent attack, we generate Snort based signature. For this, we take the payload stream of all occurrences of the attack, find the longest common subsequence and represent it in the form of regular expression. On the basis of header information and regular expression, we write Snort rule as:

$$action\ protocol\ Source\ IP : Port \rightarrow$$
$$Destination\ IP : Port\ (msg : \text{“}Message\ to\ display\text{”}$$
$$pcre : [(< regex > | m < delim >< regex >$$
$$< delim >)\ ismxAEGRUBPHMCOIDKYS]$$ [23].

After generating signature, we verify it on normal connection. If no match found then we accept it. If it generates more number of false alarms then we discard it.

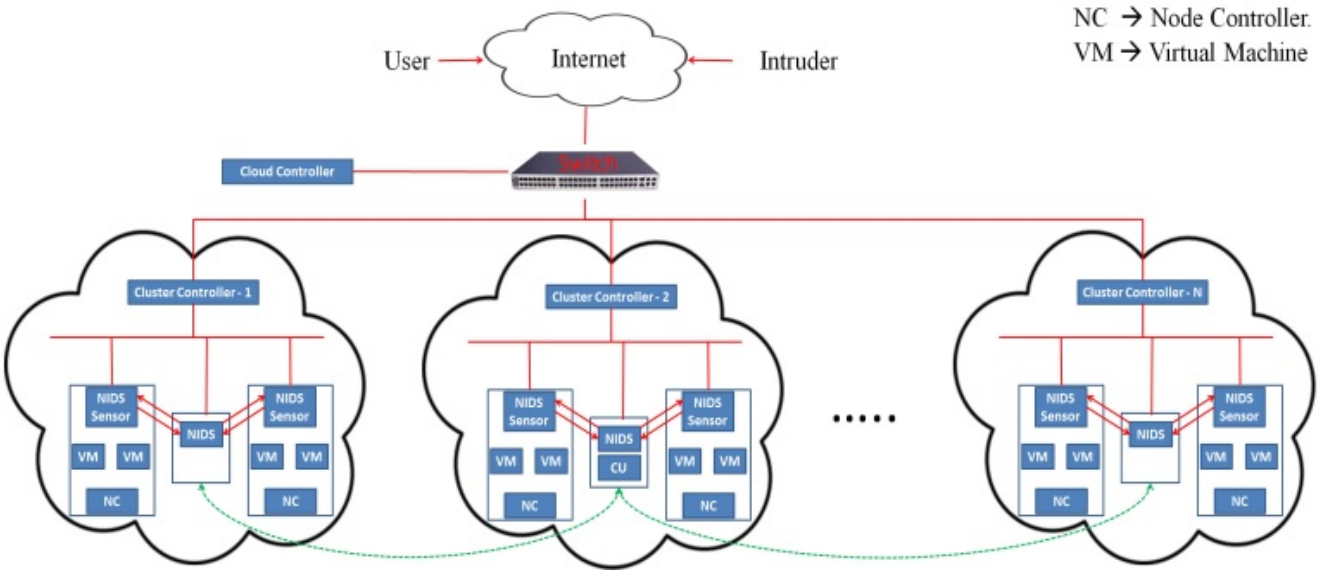

Figure 6: Experimental setup

5 Evaluation and Results

5.1 Experimental Setup

We installed eucalyptus 3.2.0 [3] cloud on CentOS 6.3. Cloud controller is on separate machine. There are $N(=3)$ cloud clusters. Each cluster contains multiple numbers of node controllers with multiple virtual machines running on each node. NIDS sensors are placed in all the Node controllers on the virtual bridge (br0) so that it can capture the internal traffic (i. e. VM-to-VM, VM-to-User etc.). We place the central database and remaining part of NIDS on a separate machine connected with the cluster. Only Node Controllers are allowed to access this machine. Correlation Unit is there in Cluster-2 as shown in Figure 6.

We use tcpdump and libpcap [26] sniffer to capture the packets. To train SVM, we use libsvm [2]. We use RBF kernel with $gamma = 0.125$ and $C = 2.0$. Window size $t = 2$ second. $S_T = 0.5$.

For evaluating performance results, we have used parameters viz; Intrusion Detected, Intrusion Missed, True Alarms, False Alarms, Accuracy, Learning and Detection time.

5.2 Results and Discussion

Evaluation of our anomaly detection system is carried on different datasets viz; KDD99 [11], NSL-KDD [21] and ITOC [10]. Details of these datasets and experiments are shown in Table 1 and Table 2.

Figure 7 shows the model generated after learning from the kddcup10% dataset. The time taken in learning is 46.616 seconds. There are 22 internal nodes, 108 leaf nodes with class label and 35 SVM models are created with the maximum height of tree is 4.

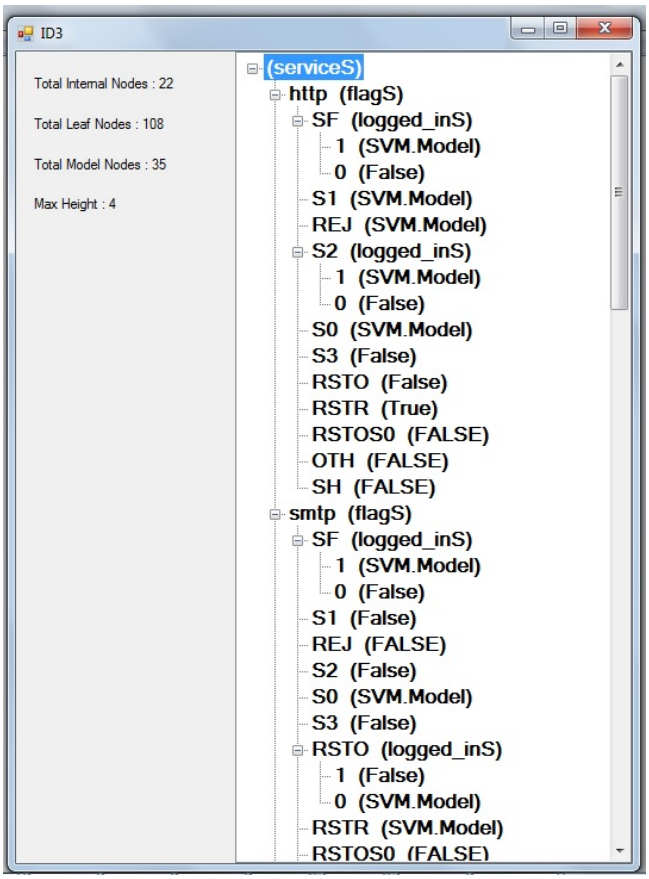

Figure 7: Screen shot of tree model generated after learning

Table 1: Details of the dataset

Training Dataset	Total Records	Intrusive Records	Normal Instances	No. of Attributes
KDD99 (10%)	4,94,021	3,96,743	97,278	41
KDD99	48,98,432	39,25,650	9,72,781	41
KDD99test (10%)	3,11,029	2,50,436	60,593	41
NSL-KDD	1,48,517	71,462	77,055	41
NSL-KDDtest	22,544	12,832	9,712	41
ITOC	4,00,000	1,67,879	2,32,121	27
ITOCtest	2,31,831	92,848	1,38,983	27

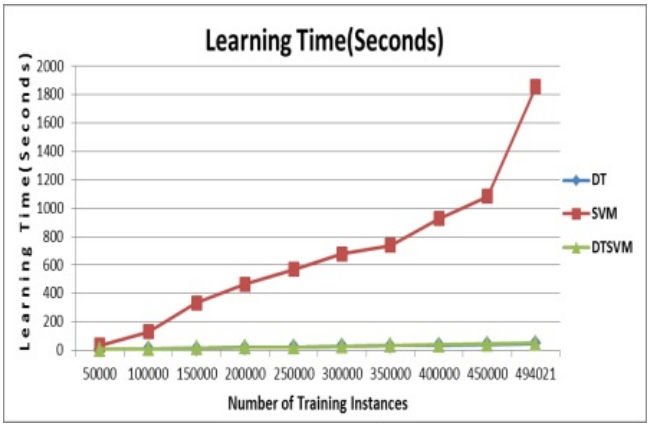

Figure 8: Comparison of learning time

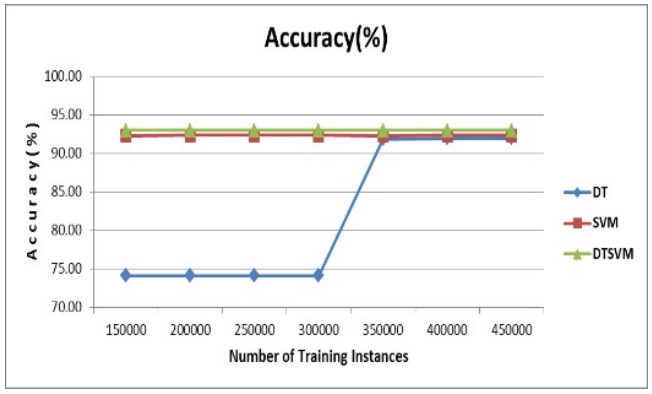

Figure 10: Comparison of accuracy

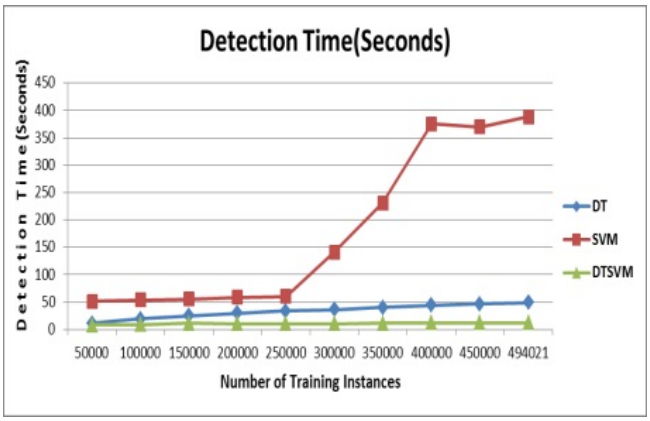

Figure 9: Comparison of detection time

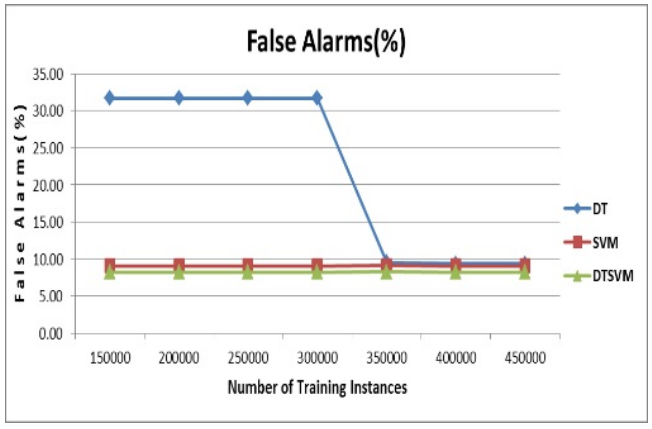

Figure 11: Comparison of false alarms

Table 2: Details of experiments

Test No.	Training Dataset	Test Dataset
Test 1	NSL-KDD	NSL-KDDtest
Test 2	KDD99(10%)	KDD99
Test 3	KDD99(10%)	KDD99test(10%)
Test 4	ITOC	ITOCtest

Figures 8, 9, 10, 11 show the behavior of decision tree, SVM and proposed ADS when we change the size of training dataset. For this we take training profiles from KDD99 (10%) and evaluate the KDD99test (10%). Figure 8 shows that learning time for the proposed ADS is almost equal to decision tree and much less than SVM. While as shown in Figure 9 the detection time is less in comparison to decision tree and SVM. Figure 10 shows that accuracy is higher than decision tree and SVM, while producing low false alarms as in Figure 11. Thus it outperforms both SVM and decision tree in terms of accuracy and computation time. Figure 12 shows the results of all the experiments listed in Tables 3 & 4 and their weighted average. Results on NSL-KDD (Test1) shows that 98.35% intrusions are detected, 1.65% intrusions are missing, 2.97% alarms are false and overall accuracy is 97.38%. Results on KDD99 (Test2) shows that 99.56% intrusions are detected, 0.44% intrusions are missing, 8.22% alarms are false and overall accuracy is 93.05%. Results on KDD99 (Test3) shows that 99.99% intrusions are detected, 0.01% intrusions are missing, 0.01% alarms are false and overall accuracy is 99.99%. Results on ITOC (Test4) shows that 86.84% intrusions are detected, 13.16% intrusions are missing, 28.34% alarms are false and overall accuracy is 84.30%. Weighted average results shows that detection time is 55 microseconds, 99.40% intrusions are detected, 0.60% intrusions are missing, 1.69% alarms are false and overall accuracy is 98.92%.

Table 3: Comparison of accuracy and detection rate

	Accuracy (%)	Detection Rate (%)
Multi SVM [14]	92.050	-
CT-SVM [12]	69.800	-
Decision Tree [16]	96.710	96.250
FER [16]	75.000	-
SVM [19]	-	98.630
Ripper Rule [19]	-	98.690
Decision tree [19]	-	98.750
DT+SVM	**98.92**	**99.40**

6 Conclusions

In proposed CIDS, cascading decision tree and SVM has improved the detection accuracy and system performance as they remove the limitation of each other. Use of DT makes the learning process speedy and split the dataset into small sub datasets. Use of SVM on each sub dataset reduce the learning time of SVM and overcome the overfitting and reduce the size of decision tree to make the detection faster. Collaboration between NIDSs prevents the coordinated attacks against cloud infrastructure and knowledge base remains up-to-date. We have performed experiments to detect the accuracy of our proposed approach with well-known KDD dataset and found encouraging results.

References

[1] B. Borisaniya, A. Patel, D. R. Patel, and H. Patel, "Incorporating honeypot for intrusion detec-tion in cloud infrastructure," in *Trust Management VI IFIP Advances in Information and Communication Technology*, pp. 84–96, Surat, India, May 2012.

[2] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[3] Eucalyptus, *Eucalyptus Website*, Sept. 27, 2015. (`http://www.eucalyptus.com`)

[4] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "A novel method for supervised anomaly detection by cascading k-means clustering and ID3 decision tree learning methods," *IEEE Transactions On Knowledge and Data Engineering*, vol. 19, no. 3, pp. 345–354, 2007.

[5] H. Garcia-Molina, "Elections in a distributed computing system," *IEEE Transactions on Computers,*, vol. 31, no. 1, pp. 48–59, 1982.

[6] J. Han and M. Kamber, *Data Mining Concepts and Techniques (2nd edition)*, San Francisco, CA: Morgan Kauf-mann Publishers, 2006.

[7] Li C. Huang and M. S. Hwang, "Study of an intrusion detection system," *Journal of Electronic Science and Technology*, vol. 10, no. 3, pp. 269–275, 2012.

[8] K. Hwang, M. Cai, Y. Chen, and M. Qin, "Hybrid intrusion detection with weighted signa-ture generation over anomalous internet episodes," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 1, pp. 41–55, 2007.

[9] K. Hwang, Y. Chen, and H. Liu, "Defending distributed systems against malicious intrusions and network anomalies," in *Proceedings of 19th IEEE International Symposium on Parallel and Distributed Processing*, Denver, Colorado, Apr. 2005.

[10] ITOC, *ITOC*, Sept. 27, 2015. (`https://www.itoc.usma.edu/research/dataset/`)

Table 4: Evaluation results

	Intrusion Detected(%)	**Intrusion Missed(%)**	**True Alarms(%)**	**False Alarms(%)**	**Accuracy (%)**
Test1	98.35	1.65	97.03	2.97	97.383
Test2	99.56	0.44	91.78	8.22	93.050
Test3	99.99	0.01	99.99	0.01	99.988
Test4	86.84	13.16	71.66	28.34	84.30
Wt. Avg.	99.40	0.60	98.31	1.69	98.92

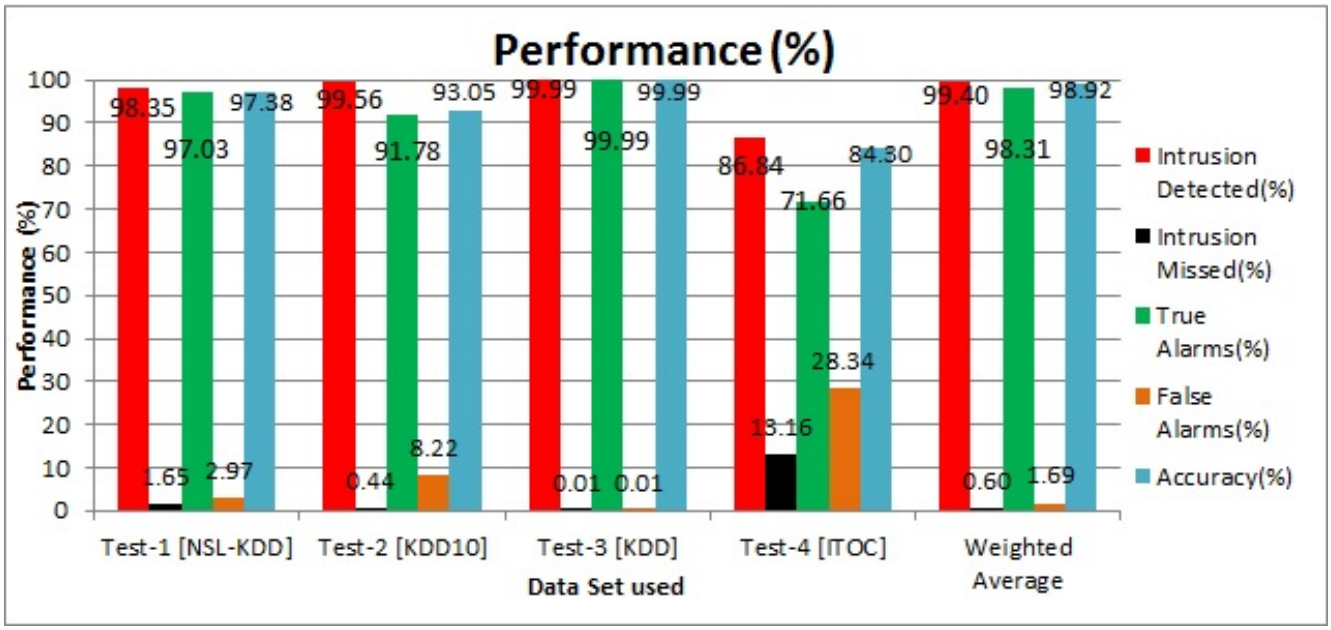

Figure 12: Evaluation results as per Tables 3 & 4 and their weighted average

[11] KDD, *KDD Cup 1999 Webpage*, Sept. 27, 2015. (`http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`)

[12] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB Journal*, vol. 16, no. 4, pp. 507–521, 2007.

[13] C. C. Lo, C. C. Huang, and J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," in *39th International Conference on Parallel Processing Workshops*, pp. 280–284, San Diego, CA, Sep. 2010.

[14] A. Mewada, P. Gedam, S. Khan, and M. U. Reddy, "Network intrusion detection using multiclass support vector machine," *International Conference on ACCTA*, vol. 1, no. 2, pp. 2, 2010.

[15] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intru-sion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.

[16] C. Modi, D. Patel, B. Borisanya, A. Patel, and M. Rajarajan, "A novel framework for intrusion detection in cloud," in *Proceedings of the Fifth International Conference on Security of Information and Networks*, pp. 67–74, Jaipur, India, Oct. 2012.

[17] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1702–1707, Honolulu, HI, May 2002.

[18] An na Wang, Y. Zhao, Y. T. Hou, and Y. L. Li, "A novel construction of svm compound kernel function," in *International Conference on Logistics Systems and Intelligent Management*, pp. 1462–1465, Harbin, Jan. 2010.

[19] R. C. A. Naidu and P. S. Avadhani, "A comparison of data mining techniques for intrusion detection," in *IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT'12)*, pp. 41–44, Ramanathapuram, Aug. 2012.

[20] R. M. Needham, "Denial of service: an example?," *Communications of the ACM*, vol. 37, no. 11, pp. 42–46, 1994.

[21] NSL, *The NSL-KDD data set*, Sept. 27, 2015. (`http://nsl.cs.unb.ca/NSL-KDD/`)

[22] G. Paliouras and D. S. Bree, "The effect of numeric features on the scalability of inductive learning programs," in *Proceedings of the European Conference in Machine Learning*, pp. 218–231, Crete, Greece, Apr. 1995.

[23] M. Roesch and C. Green, *Snort User?s Manual 2.9.3: The Snort Project*, Technical Report 2.9.3, May 2012.

[24] S. V. Sandar and S. Shenai, "Economic denial of sustainability (EDOS) in cloud services using http and xml based ddos attacks," *International Journal of Computer Applications*, vol. 41, no. 20, pp. 11–16, 2012.

[25] Snort, *Snort Website*, Sept. 27, 2015. (`http://www.snort.org`)

[26] Tcpdump, *Tcpdump and libpcap*, Sept. 27, 2015. (`http://www.tcpdump.org/`)

[27] M. Xu, J. Li Wang, and T. Chen, "Improved decision tree algorithm: ID3+," in *Intelligent Computing in Signal Processing and Pattern Recognition Lecture Notes in Control and Information Sciences*, pp. 141–149, Crete, Greece, Aug. 2006.

[28] V. Yasami, S. Khorsandi, S. P. Mozaffari, and A. Jalalian, "An unsupervised network anomaly detection approach by k-means clustering & ID3 algorithms," in *IEEE Symposium on Computers and Communications*, pp. 398–403, Marrakech, July 2008.

[29] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, 2010.

Dinesh Singh is currently pursuing the Ph.D. degree in Computer Science and Engineering from Indian Institute of Technology Hyderabad, India. He received the M. Tech degree in Computer Engineering from the National Institute of Technology, Surat, India, in 2013. He received B. Tech degree from R. D. Engineering College Ghaziabad, India, in 2010. He joined the Department of Computer Science and Engineering, Parul Institute of Engineering and Technology Vadodara, India as an assistant professor from 2013 to 2014. His research interests include machine learning, big data analytics, visual computing, cloud computing, intrusion detection.

Dhiren Patel is currently a professor in Computer Engineering Department at NIT Surat, India. He leads Security and Cloud computing group at NIT Surat. His research interests include Information Security, Cloud Computing & Trust Management, Internet of Things and Green IT. Prof. Patel has academic and research associations with IIT Gandhinagar (Visiting Professor/Adjunct Professor), with University of Denver USA (Visiting Professor), with City University London (Visiting Scientist - Cyber Security), with British Telecom UK (Visiting Researcher - Cloud Security and Trust), and with C-DAC Mumbai (Research Advisor - Security and Critical Infrastructure Protection). He has authored a book on Information Security (published by Prentice Hall in 2008) and numerous research papers.

Bhavesh Borisaniya is currently pursuing PhD from the Department of Computer Engineering at National Institute of Technology, Surat, India. His research interests include security in cloud computing and virtualization, intrusion detection system, and honeypot.

Chirag Modi is currently working in Computer Science and Engineering at National Institute of Technology Goa. He holds Ph. D (2010-2014) and M. Tech (2008-2010) in Computer Engineering from National Institute of Technology, Surat. Dr. Modi's research interests include security, privacy, data mining and cloud computing with primary focus on intrusion detection in cloud computing and privacy preserving data mining. Apart from contributing in various internal conferences, workshops and training programs, Dr. Modi has published many papers in reputed SCI journals and international conference proceedings. He is an active researcher in Computer Science field, and acting as a TPC member, Editor and Reviewer in many reputed international conferences as well as journal. In addition, he is frequently delivering an expert talk at many institutes and also explores many research areas.