A feasible approach to intrusion detection in virtual network layer of Cloud computing

CHIRAG MODI^{1,*} and DHIREN PATEL²

¹National Institute of Technology Goa, Farmagudi 403 401, India

²National Institute of Technology Surat, Surat 395 007, India
e-mail: cnmodi@nitgoa.ac.in

MS received 12 December 2015; accepted 6 March 2018; published online 22 June 2018

Abstract. Intrusion detection/prevention is the greatest security challenge at virtual network layer of Cloud computing. To address this challenge, there have been several security frameworks reported. However, still there is a scope of addressing newer challenges. Here, we propose a security framework to detect network intrusions in Cloud computing. This framework uses *Snort* and combination of different classifiers, viz *Bayesian*, *Associative* and *Decision tree*. We deploy our intrusion detection system (IDS) sensors on each host machine of Cloud. These sensors correlate intrusive alerts from each region of Cloud in order to identify distributed attacks. For feasibility analysis and functional validation of this framework, we perform different experiments in *real time* and *offline* simulation.

Keywords. Intrusion detection; network security; Cloud computing; virtualization; classifier.

1. Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage and applications), which can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. With this mode, Cloud users are able to communicate with Cloud via *front end* (see figure 1). Cloud *back end* consists of computer hardware (processing servers/host machines) and software, which are managed to deliver different services. Through *back end*, the user's requests are processed in order to allow them to access virtual machine (VM) instances. VMs are interconnected and communicate to each other through virtual network, where network traffic among VMs is routed using virtual switches (vSwitches). Such a traffic can primarily use virtualization protocols (e.g., VLAN protocol).

As the number of attacks continues to rise, organizations are concerned about the safety of moving their computational assets to Cloud. As per survey done by Popovic and Hocenski [2] and International Data Corporation (IDC), security is the major concern of Cloud [3]. Biggs and Vidalis [4] stated that 50% of approximately 170 businesses concern about the security of Cloud resources. Cloud infrastructure is shared by potentially millions of users. By abuse and nefarious use of the shared infrastructure, attackers exploit vulnerabilities of the Cloud. They may

install vulnerable software on their VMs, which may create loopholes in Cloud security. In Cloud, malicious user having a VM instance can perform various attacks on other's VM. Researchers [5, 6] have investigated Cloud security concerns at different layers, viz application, network, data storage, virtualization, authentication and access control, trust and regulatory compliance. A survey by Lockheed Martin [7] from Cyber Security division declares that intrusion detection in the Cloud is a major security concern after data security.

Looking into Cloud networks (see figure 1), there are mainly two security threats, viz *External attacker* and *Insider attacker*. External network attackers often perform DoS/DDoS attacks to affect the availability of Cloud resources as well as services. In addition, other traditional attacks such as scanning, spoofing, DNS poisoning and phishing attack are performed for unauthorized access of Cloud resources. Insider attacker (attacker within the Cloud network) could be a malicious user at client side, malicious user at provider side or Cloud provider itself. They can learn authentication information to gain access of the other's VMs. Major attacks in this category are VM-to-VM and VM-to-host machine attacks. Gupta *et al* [8] have presented Cloud-network-specific attacks, viz DoS attack, TCP session hijacking, DNS attacks, Network penetration, Fragmentation attack and Eavesdropping.

To secure Cloud networks, traditional network security mechanisms such as firewall and intrusion detection system (IDS) face new challenges due to the following. In Cloud, a large number of VMs produce heavy network traffic. Due to

*For correspondence

Figure 1. General architecture of Cloud computing and threat model.

the virtualized and distributed nature of the Cloud, it is difficult to detect distributed attacks. In Cloud, VM instances are provisioned or de-provisioned dynamically; therefore, IDS has to scale up or down accordingly.

This paper demonstrates the proposed security framework named as Hybrid Network Intrusion Detection System (Hybrid-NIDS) for Cloud. We deploy Hybrid-NIDS sensors on each host machine (node) of Cloud. These sensors are made of *Snort* and combination of different classifiers, viz *Bayesian*, *Associative* and *Decision tree*. Sensors on each host machine monitor network traffic of VMs and host machine for intrusions. After detecting intrusion, it sends alert of intrusion to *central log* for correlation. *Central log* determines whether that alert belongs to distributed attack or not. We validate feasibility of our Hybrid-NIDS for Cloud by performing different experiments on a Cloud test bed. We analyse the Cloud IDS requirements, which are fulfilled by the proposed security framework.

The organization of this paper is as follows. Section 2 surveys Cloud IDS frameworks with further research scope. The proposed security framework is discussed in section 3. Performance evaluation and analysis of the proposed framework are presented in section 4. Section 5 concludes this research work with references at the end.

2. Cloud IDS: literature survey

2.1 IDS in Cloud (Cloud IDS): challenges

Intrusion detection involves the process of monitoring network (or system) for malicious activities such as executing malware, unauthorized access to system, attempt to gain higher privileges, etc. IDS attempts to detect an intruder breaking into system or a legitimate user misusing system resources. Although capability of IDS has been tested to work well in some large scale networks, its deployment and utilization in Cloud is a challenging task [9]. As per Cox [10], IDS can be deployed on each VM, each node (host machine or hypervisor) or in traditional

network. Deploying IDS on each VM allows monitoring the activities of VM itself. However, it increases the complexity of VM-IDS management since VMs are dynamically added into Cloud. Deploying IDS on each host machine allows monitoring the host machine's activity, underlying hypervisor's activity and anything travelling between the VMs on that hypervisor. This helps monitoring virtual network (the network established within the host itself). However large amount of data communication (between VMs) reduces performance of IDS and may cause packet dropping at IDS. Therefore, IDS should be fast, if it is deployed on host machine to monitor multiple VMs. IDS can be deployed in traditional network (external network of Cloud) that is an entry point of Cloud system. This allows detecting external attacks to Cloud. However, it cannot help in detection of internal attacks.

Cloud IDS has some additional challenges [11]. In virtual environment of Cloud, switch is virtualized (on each host system) to allow communication between VMs, and therefore, there is a need of monitoring network traffic that flows through virtual switch. In Cloud, VMs are dynamically added (or removed). In addition, security requirements for each VM may be different. Therefore, signature database (or behaviour database) should be dynamic and relevant to each VM. As different attacks, their complexity and unpredictability increase in current network technology, IDS should be trained to detect new attacks with the help of previously observed events. This improves its detection capability with minimum false alerts. In addition, there is a need of maintaining performance and security. Therefore, effective techniques should be employed of reducing false alerts and maintaining performance. Cloud IDS must be capable of automatically handling high data flow without human intervention. Shared and virtualization technology in Cloud increases vulnerabilities in Cloud. Some vulnerabilities [12, 13] in hypervisor allowed an attacker to expose inappropriate access and control of the VMs. Thus, these vulnerabilities allow an attacker to compromise IDS running on VMs. Techniques used in intrusion detection should have less computational cost so that Cloud IDS can work in *real time*. The deployment of IDS sensors and alert correlation module affect the scalability and performance since these modules work together in intrusion detection. Cloud IDS must protect itself from unauthorized access and attacks. It should be able to authenticate network devices and in order to protect its data.

2.2 Cloud IDS: existing approaches

Roschke *et al* [9] have presented a VM-IDS framework (see figure 2), which gathers malicious events identified by IDS sensor and stores in the event database. Analysis component (configured by users) accesses the event database and analyses events. IDS sensor identifies the malicious events

Figure 2. VM-compatible IDS architecture [9].

and sends them to an event gatherer. As IDS deployed on each VM filters network attacks, this approach secures VMs from malicious activities. However, with the increase of VM instances, IDS instances also increase.

A flexible method based on mobile agents (MAs) to detect intrusion for Cloud applications was proposed by Dastjerdi *et al* [14]. It uses MAs to protect VMs running outside the organization. The evidences of an attack from all the attacked VMs are collected by MA for further analysis and auditing. Thus, this approach is suitable to identify intrusions in VM migrating outside the organization. However, if the number of VMs (attached to MA) increases, it produces more network traffic to MA.

In other agent-based approaches by Lo *et al* [15] and Ram [16], individual IDS sensor is deployed at each Cloud region (see figure 3), which detects intrusion and drops attacker's packet. Then, an alert of the detected intrusion is sent to other regions. At each region, alerts generated by other regions are collected by an alert clustering module, and the correctness of that alert is decided by calculating its severity. This approach prevents Cloud from DDoS attack. In this approach, each sensor sends intrusion alert to all the sensors. Thus, if there are n sensors detecting intrusion, then $n \times (n-1)$ alert messages will be exchanged. Thus,

Figure 3. Agent-based IDS architecture [15].

communication overhead will be increased with increasing number of sensors.

Bakshi and Yogesh [17] have installed IDS in virtual switch in order to detect DDoS attack in a VM. Here, IDS in virtual switch monitors/logs virtual network traffic into the database. These logged packets are matched with known signatures for any correlation in order to identify intrusions, and then the nature of attacks is notified to the virtual server. Later, packets from the specified IP are dropped by the virtual server, and if there is a DDoS attack, all the *zombie* machines (that are victims of DDoS attack) are blocked. Thus, DDoS attack can be blocked by this approach. It detects only known attacks.

Mazzariello *et al* [18] have deployed *Snort* at the Cloud controller as well as on physical machines (hosting VMs) in open source eucalyptus Cloud in order to detect intrusions. This deployment solves the problem of creating multiple instances of IDS. It is a cost-effective solution only for known attacks.

Third-party-based Cloud IDS framework was proposed by Gul and Hussain [19]. In this framework, Cloud IDS monitors network traffic (incoming to infrastructure) at *front end* for malicious packets and sends intrusion alerts to a third party monitoring service. Third party monitoring service provides instant report (about the intrusion) to Cloud user organization management system and also sends advisory report (about loophole) to Cloud service provider (CSP). This approach uses multi-threading technique to improve detection speed. However, there is a need of detecting attacks in virtual network.

In Dhage's IDS framework [20], IDS controller (centralized) provides instance of mini-IDS to user, whenever Cloud services are accessed. Mini-IDS instance monitors user's activities (network traffic), and sends logs of session to IDS controller. When a user starts a session, IDS controller checks the knowledge base, containing information (patterns) of user activities. Knowledge base uses artificial neural network that is trained using user's previous activities and generates user's activity patterns. With the help of activity patterns of user (stored in knowledge base), IDS controller identifies anomalous behaviour of user. Performance results of IDS controller are not reported yet.

CIDS-Cloud based IDS [21] has two IDS detectors on each node, CIDS and HIDS. The node cooperatively participates in intrusion detection by identifying security violations and its audit data with other nodes. Here, sensor collects system logs, audit data and sequence of user actions and commands from each VM and sends to CIDS and HIDS detectors that implement intrusion detection outside the VM (at node).

Network IDS based on profiles for detecting intrusions in virtual environment was proposed by Gupta *et al* [8]. Here, each VM is analysed for patterns of attack based on the behaviour of network traffic. The malicious behaviour is then sent to the alert and response generation component.

The alert and response component generates alert and sends it to the administrator for further analysis. There is a need of detecting distributed attacks.

In a Cloud IDS framework by Idrees *et al* [22], network intrusions are detected at perimeter network point in the DMZ zone, and system level attacks are detected in the critical machines (or servers). IDS is integrated with *Snort*-based signature detection module, and classification-and clustering-based anomaly detection module. Here, the captured network traffic is parsed into multiple threads for concurrent execution and sent to the queue handler. The data packets from the queue handler are analysed by signature-based technique and anomaly technique. Intrusion alerts are sent to the supervisor unit for preparation of alert reports.

We identify research gaps as follows. Existing proposals [15–19] deploy NIDS sensor at an entry point of Cloud and identify only external intrusions. This will not help detect internal attacks in virtual network of Cloud. Some proposals by [9, 15–19] detect only known attacks in virtual network since they use only signature-based technique. There is a need of detecting unknown attacks. Cloud IDS frameworks from [8, 20–22] detect unknown attacks in virtual network; however IDS should be fast to inspect high volume of traffic from a large group of VMs in Cloud. Otherwise, severe packet dropping may occur at the sensor. Approaches [9, 14] deploy their IDS sensors on each VM in order to protect VMs. Deployment of IDS at each individual VM creates additional loophole for host manipulation attack. The host can subvert the individual IDS or sensors deployed at any VM, making it difficult to detect intrusions [11]. There is a need of providing resistance capability to IDS (protection from being compromised). Most of the proposals [8, 14, 17–22] do not correlate intrusion alerts from multiple nodes in order to identify distributed attacks. In addition, properties applicable to traditional IDS (viz better true positive rate, low false alerts, high accuracy) need to be researched for Cloud IDS.

3. Hybrid-NIDS: proposed security framework

The objective of this work is to develop an IDS for Cloud computing environment, which can detect intrusions in virtual network with high efficiency. It should have less number of false alerts. It should correlate the evidences (of alerts) in the entire network with minimum communication overhead. The elaborated design goals are as follows.

Hybrid-NIDS should detect the network intrusions from external and internal network traffic to secure each component (e.g., host machine, VMs, traditional network, etc.) of Cloud. This has to be achieved by monitoring network traffic at physical network layer and virtual network layer.

In addition, it should detect distributed attacks. It should be fast to detect attacks (e.g., without dropping packets at sensors) in heavy traffic of Cloud. We deploy Hybrid-NIDS sensors on each host machine in Cloud. Thus, Hybrid-NIDS is capable of monitoring traffic from external network to VMs and VM to VM (from virtual switch). Such deployment helps monitor multiple VMs (on the same host) at a time, and to secure host machine and VMs from network intrusions. In addition, Hybrid-NIDS sensors (on each host machine) send intrusion alerts to central log, which helps identify distributed attacks. Hybrid-NIDS should be accurate and capable of detecting true intrusions. Hybrid-NIDS should be scalable as VMs are added or removed. In addition, it should not be compromised by an attacker.

3.1 Background: hybrid approach

The motivation of using hybrid approach is to improve the detection capability of IDS by combining signature-based detection technique and anomaly detection technique. Therefore, we have combined signature-based detection (using *Snort* and *Signature Apriori*) and anomaly-based detection (using *Bayesian*, *Associative* and *Decision tree* classifiers).

Snort [23] is an open source tool that works based on signature detection technique. It monitors network data packets, pre-processes them and matches their contents with the known attack patterns (using *Snort* rules) for any correlation. *Signature Apriori* [24] takes captured packets and known patterns as an input, and generates derivative of known attack signatures as an output. These signatures can help identify derivative of known attacks. In general, it derives longest common frequent patterns in network packets.

Input: Network packet payload dataset D , known attack patterns (A)

Output: Derived signatures D_S

Algorithm: Signature Apriori

1. Find frequent-1 patterns from D
 2. For each attack pattern in A
 3. For each frequent-1 pattern
 - a. Merge frequent-1 item from D to RHS of A (A_F)
 - b. If A_F is frequent

Put into the possible signature (P_S)
 4. Continue this loop till frequent sets are generated
 5. For each frequent-1 item
 - a. Merge frequent-1 item from D to LHS of P_S (F_{P_S})
 - b. If F_{P_S} is frequent

Put into the possible signature (P_S)
 6. Continue this loop till frequent sets are generated
 7. For all signatures in P_S
 - a. Put longest signature into derived signature (D_S)
-

Table 1. Performance of different classifiers in intrusion detection (w.r.t. NSL-KDD dataset [25]).

Classifier category	Classifier	Class label	True prediction (%)	False prediction (%)	Accuracy (%)	Learning time (s)
Trees	Decision tree (ID3)	Normal	99.10	1.2	98.03	0.91
		Anomaly	98.80	0.9		
	Decision tree (C4.5)	Normal	98.00	1.4	98.30	0.50
		Anomaly	98.60	2.0		
Bayes	Naive Bayes	Normal	95.10	4.7	95.19	0.13
		Anomaly	95.30	4.9		
	Bayes net	Normal	95.10	4.7	95.20	0.20
		Anomaly	95.30	4.9		
Function	RBF network	Normal	95.00	3.4	95.80	0.59
		Anomaly	96.60	5.0		
	LibSVM	Normal	98.90	1.8	98.54	171.20
		Anomaly	98.20	1.1		
Rules	Decision table	Normal	97.30	2.1	97.63	20.12
		Anomaly	97.90	2.7		
	JRip (Ripper)	Normal	97.70	2.3	97.69	92.07
		Anomaly	97.70	2.3		
	Associative classifier	Normal	85.16	0.25	95.93	0.67
		Anomaly	99.75	14.84		

For anomaly detection, we use three different classifiers, viz *Bayesian*, *Associative* and *Decision tree*. A classifier is a model that is built from a set of labelled data (called training set) and that model is used to predict the class value of new data (for which the class label is unknown) from other attributes (or features).

The reason for choosing these three classifiers (Bayesian, Decision tree and Associative classifier) is that they have better performance results on existing network dataset (see table 1). As shown in table 1, *Decision tree classifier* is more accurate than the other well-known classifiers, while *Bayesian classifier* has less learning time. Therefore, we have chosen these two classifiers in Hybrid-NIDS. However, we observed that these two classifiers produce more false alerts than *Associative classifier*. Therefore, we have combined *Associative classifier* with *Decision tree* and *Bayesian classifiers*. In addition, these three classifiers have low detection time compared with other classifiers. Thus, these three classifiers seem to fit very well for Cloud IDS. Taking different advantages of these classifiers, we can come up with accurate final decision of intrusive alert.

Bayesian classifier derives the conditional probability of a given profile to a particular class [26]. For anomaly detection, it works as follows: let D be a set of network traffic profiles with class labels, and each profile is represented by a feature vector $F = (f_1, f_2, \dots, f_n)$. Consider m different classes $C_1, C_2, \dots, C_m$. Now, *Bayesian classifier* classifies unknown profile F to class C_i , if the conditional probability $P(C_i|F)$ is the highest among all the $P(C_m|F)$, for all the m classes. Thus, the objective

is to derive the maximal $P(C_i|F)$. It can be derived as follows:

$$P(C_i|F) = (P(F|C_i)P(C_i))/P(F)$$

$P(C_i|F) \approx (P(F|C_i)P(C_i))$, since $P(F)$ is constant for all classes.

Thus, *Bayesian classifier* classifies class label (normal or intrusion) of the given network traffic profile by observing previously stored events (network traffic profiles).

Associative classifier explores highly confident associations among multiple features [27]. For Hybrid-NIDS, we use it as follows. Let D be a dataset of network traffic profiles that has n features $(f_1, f_2, \dots, f_n)$ and one class feature $C(C_1, C_2, \dots, C_n)$. *Associative classifier* generates rules in the form of $R: F \rightarrow C$, where F represents condition set of $(F, value)$ pairs and C as a class label. The generated rules are then used to predict the class label of unknown profile. If two different rules on different sets of features have different predictions, then the rule having higher confidence is considered for prediction. *Associative classifier* can help detect DoS/DDoS attack efficiently since it uses the frequency of attack patterns.

Decision tree classifier generates a tree structure, where each node indicates a feature name, each leaf represents a class label and each branch indicates an outcome of the associated node. It has faster learning speed than those of other classification techniques and can classify large amount of data [27]. For anomaly detection, we include ID3, a decision tree classifier [28] in our Hybrid-NIDS. Steps for building a decision tree are as follows:

Input: Network traffic profiles D .

Output: Decision tree T .

Step 1:

- If all the network traffic profiles in D belong to Normal class, then create a Normal class node and stop.
- If all the network traffic profiles in D belong to Intrusion class, then create an Intrusion class node and stop.
- Otherwise select a feature F (with different values $f_1, f_2, \dots, f_n$) of the network traffic profiles and create a decision node.

Step 2:

- Partition the network traffic profiles in D into subsets $D_1, D_2, \dots, D_n$ according to different values of F .

Step 3:

- Apply the algorithm recursively to each of the sets D_i .
-

For selecting a feature from network traffic profiles in order to split in the decision tree, gain for all the features is calculated. Then, a feature having the highest gain is taken for splitting. Gain of each feature is derived using Eqs. (1)–(3):

$$Info(D) = - \sum_{i=1}^m P_i \times \log_2(P_i) \quad (1)$$

where m is the number of classes, P_i is the probability that a profile in D belongs to class C_i , calculated by $|C_i, D|/|D|$.

$$Info_F(D) = - \sum_{j=1}^v (|D_j|/|D|) Info(D_j), \quad (2)$$

$$Gain(F) = Info(D) - Info_F(D). \quad (3)$$

Equation (2) indicates that the information needed after using F to split D into f splits, whereas Eq. (3) represents information gained by branching on feature F . Here, each path from root to leaf indicates classification rule. These rules help predict class label of unknown profile.

3.2 System architecture of our framework

As shown in figure 4, we deploy Hybrid-NIDS sensors on each host machine of Cloud. Here, number n for Cloud nodes can be a large number. Each sensor captures the network traffic (incoming/outgoing) to VMs and host machine, and then detects intrusions. After detecting intrusions, it sends alert of intrusion to *central log* for correlation. *Central log* determines whether that alert belongs to distributed attack or not. For security reason, *central log* is accessed only by sensors.

Proposed Hybrid-NIDS sensor consists of modules, viz *Packet capture*, *Signature-based detection*, *Network traffic feature extractor*, *Anomaly detection*, *Score calculation* and *Alert system* (see figure 5).

3.2a Packet capture: Using packet sniffing library (e.g., libpcap [29]), Hybrid-NIDS sensor (deployed on each host machine) captures the network traffic (incoming/outgoing) to VMs and host machine. These packets are inspected in *real time* by *signature-based detection* technique.

3.2b Signature-based detection: For *signature-based detection*, we use knowledge base and the derived attack rule base. Knowledge base stores known attack rules. To generate derived attack rules, we use a *Signature Apriori* algorithm (*offline*) that takes the captured packets and known attack rules as an input, and generates derived attack rules, as presented earlier. Derived attack rules are stored in the derived attack rule base. Here, *Snort* checks these packets with the rules in knowledge base and derived attack rule base to find any correlation. If it finds any packet with intrusion, it determines the nature of attack and sends alert to *score calculation*. Captured packets are then applied to the *network traffic feature extractor*.

3.2c Network traffic feature extractor: It generates network traffic profiles from the captured network packets. As shown in figure 6, it stores each packet frame in an ongoing connection cache. A connection is a sequence of packets flowing to and from source to target under some well-defined protocol and at a certain time. When any connection is completed (e.g., closed, reset or terminated connection), it is stored in a completed connection list. For each completed connection, network profile is generated by extracting the network features, viz *basic features*, *traffic-related features* (network traffic statistics), *content* (payload) and *virtualization-related features* from the raw packets.

Basic features involve all the attributes that are extracted from a TCP/IP connection, e.g., protocol, service, size of traffic flow, etc. *Content features* involve inspecting data portions of the packets for respective connection. To detect attacks (e.g., remote to local and user to root) that are embedded in data portions of the packets, suspicious behaviour in the data portion is looked at within a given time frame t , e.g., number of failed login attempts, no of root access, etc. *Traffic features* are derived within time frame t , and divided into two groups, viz same host features and same service features. Same host features involve the connections having same destination host within a given time frame (e.g., 2 s) and statistics related to protocol, service, flag error, etc. Same-service features include the connections having same services within given time frame and calculate traffic-related statistics. A detailed description of each feature is given by KDD [30]. In addition, Virtual LAN (VLAN) ID is derived from Ethernet frame to identify VLAN number where malicious VM is running.

Figure 4. Deploying Hybrid-NIDS sensors in Cloud.

3.2d Anomaly detection: For anomaly detection, we use three different classifiers, viz *Bayesian*, *Associative* and *Decision tree*. These three classifiers are trained using previously observed network traffic profiles that are stored in a behaviour base named as network traffic profile base. As discussed earlier, they individually predict the class label (intrusion or normal) of the given network traffic profile and send their results to *score calculation* for final decision making.

3.2e Score calculation: It determines whether the intrusion detected by any classifier is detected as intrusion by other classifiers or not. We tested different fusion techniques, viz *Detect all method*, *Majority voting method* [31, 32] and *Weighted averaging method* [32] for final decision making. These techniques help improve detection accuracy of Hybrid-NIDS [33]. The *Detect all method* takes output from each of the classifiers in Hybrid-NIDS. Then, it predicts the given profile as an Intrusion, if all classifiers predict that profile as an Intrusion. Otherwise, it considers that profile as a normal profile. The *Majority voting method* predicts the given profile as an Intrusion if more than one classifier predict that profile as an Intrusion. In the

Weighted averaging method, weighted average of profile is derived by the relation

$$\text{weighted_average}(\text{Profile}) = \frac{\sum_{i=1}^N W_i O_i}{N}$$

where W_i is the weight assigned to i^{th} classifier, O_i is an output of the i^{th} classifier and N is the number of classifiers involved in anomaly detection.

If the weighted average of any profile satisfies the defined threshold T , then that profile is considered as an intrusion. Otherwise, it is considered to be normal. As per our observation, *weighted average method* has better performance results in identifying intrusions.

In addition, *score calculation* determines whether the detected intrusion is a distributed attack or not. For identifying distributed attack from the detected intrusions, we check the same intrusion alerts (determined either by *Snort* or *score calculation*) in *central log* (within a given time frame T_f) where intrusion alerts from sensors at other regions are collected. Whether the given alert is of distributed attack or not is determined by the *majority vote method* [31] and is derived by the equation

Figure 5. Design of the proposed Hybrid-NIDS sensor.

$$\text{majority_vote}(\text{intrusion}) = \frac{N_s}{N}$$

where N_s is the number of sensors sending same alerts to *central log* within a given time frame, and N is the number of sensors deployed in whole Cloud.

If $\text{majority_vote}(\text{intrusion}) \geq t$ (where t , e.g., 0.5, is the threshold defined by security administrator), then that intrusion is considered as distributed attack. This method can secure whole Cloud from distributed attacks. With the help of *central log*, intrusion detected at any host will be updated in knowledge base and behaviour base (network traffic profile base) of other sensors deployed on other hosts. Thus, the next time, such intrusion can be detected using *signature-based detection* technique at other hosts.

3.2f Alert system: It generates alert of intrusion that is determined either by *Snort* or by *score calculation*. Alert includes the information of intrusive connection (e.g., IP of attacker, IP of victim VM, source port, destination port, protocol and detection result). It stores the alerted intrusion in the *central log*.

In our security framework, we have applied *signature-based detection* followed by *anomaly detection*, which helps reduce detection time as anomaly detection has to identify only unknown attacks. As discussed in section 2.2, agent-based approaches [15, 16] require $n(n-1)$ messages for exchanging an alert, where n is the number of sensors. Here, each Hybrid-NIDS sensor sends an intrusion alert to the central log. Therefore, it requires only n messages to store an alert of distributed attack. If the *central log* fails, it

may affect only the detection of distributed attacks, but will not affect the detection at individual host. This limitation can be addressed as the central machine allows only host machines (where Hybrid-NIDS sensor is deployed) to access *central log*. Therefore, attackers have to first compromise the host machine in order to compromise *central log*. It requires lot of efforts to get root access of the host machine.

4. Validation and performance evaluation

4.1 Test set-up

For functional validation, we have created a test set-up at Cloud Security Research Lab (NIT Surat) with five host machines and fifteen VMs with different operating systems, viz Windows 7, Ubuntu 10.10 and Windows Server 8 (see figure 7). These VMs communicate with each other through a virtual network interface that shares host network with VMs. We have installed a Hybrid-NIDS sensor on each host machine. For performing various attacks on VM, we have used different tools and scripts [34–37], which flood malicious traffic to VMs. From the virtual network interface, the Hybrid-NIDS sensor monitors virtual network packets for intrusion and sends alert to *central log* with the help of TCP client-server socket programming [38].

We have conducted different experiments in *real time* and *offline* simulation for Hybrid-NIDS. In *real time* simulation of Hybrid-NIDS, different attacks are performed to VMs that are on the same host (see table 2) or on different

Figure 6. Network traffic features extractor.

hosts (see table 3). For creating an unknown attack environment, we trained our sensors based on a few known attacks, and other known attacks are considered as unknowns in our test environment.

In *offline* simulation, we verify performance (accuracy and validation) of Hybrid-NIDS w.r.t. different intrusion datasets, viz KDD intrusion dataset [30], DARPA dataset [39] and CAIDA dataset [40]. Although these datasets include only traditional attacks, they can be used to test the feasibility of Hybrid-NIDS in terms of Cloud IDS requirements against traditional attacks. We train classifiers in Hybrid-NIDS using these datasets (having older attacks), while recent attacks on Cloud are considered as unknown attacks, which should be detected by these trained classifiers. In training datasets, class labels are generalized into normal and intrusion classes. Table 4 shows experiments conducted in *offline* simulation of Hybrid-NIDS. In each experiment, training dataset is used to train classifiers in Hybrid-NIDS and to generate classification rules, while testing dataset is applied to classifiers for predicting the class label. Researchers [41–43] have shown that only a subset of the features are relevant to each type of attack. If

we use non-relevant features in classification, they affect the overall detection accuracy. Therefore, we have used only 17 features that have better relevance to each class label. The relevance of feature to class label is calculated using Gain [27]. This helps us to improve detection accuracy and speed.

4.2 Results and analysis

For non-distributed attacks, detection results (in *real time* simulation) of Hybrid-NIDS are shown in figure 8. It shows that attacks performed on victim VM are detected by Hybrid-NIDS at the host machine. Hybrid-NIDS detects more than 82% of intrusive connections in *real time* simulation. This implies that 18% connections are considered as normal. This is due to approximately 15.39% noise traffic (that is not relevant to victim VM) produced during simulation.

In case of distributed attacks, detection results (in *real time* simulation) of Hybrid-NIDS are shown in figure 9. Results show that Hybrid-NIDS identifies more than 87%

Figure 7. Test set-up.

Table 2. Experiments in *real time* simulation (attacks on VMs at the same host).

No.	Attacking VM	Victim VM	Attack type and tools
1	VM1-2	VM1-3	Hping, port scan, TCP-SYN flooding
2	VM1-1	VM1-3	UDP flood, Xpinger, HTTP flood (DoS), fishing port scanner, Fast port scanner, FTP scanner
3	VM1-1 and VM1-2	VM1-3	UDP flood, Xpinger, HTTP flood (DoS), Fishing port scanner, Fast port scanner, FTP scanner, Fastest host scanner, Hping, port scan, TCP-SYN flooding

of distributed attacks in *real time* simulation. As discussed earlier, we have used a centralized approach to detect distributed attacks. For this, the central server receives alerts from all the sensors deployed on host machines. Thus, we check a number of alerts dropped at the central server. As shown in figure 10, the central server drops less than 3% alerts that are sent by sensors. Although, dropped alerts are not identified as distributed attacks, they are considered as

intrusions at individual hosts. Thus, it will not affect the detection accuracy.

Performance results of Hybrid-NIDS in *offline* simulation are as follows.

Figure 11 shows the true positive rate for individual classifiers and Hybrid-NIDS on different datasets of intrusions. Ideally, true positive rate should be 100%. Hybrid-NIDS has a true positive rate greater than 99%. It shows

Table 3. Experiments in *real time* simulation (distributed attack).

No.	Attacking VMs	Victim VMs	Attack type and tools
1	VM1-2, VM2-2 and VM3-2	VM1-3, VM2-3 and VM3-3	TCP-SYN flooding
2	VM1-2, VM2-2, VM3-2 and VM4-2	VM1-3, VM2-3, VM3-3 and VM4-3	TCP-SYN flooding
3	VM1-2, VM2-2, VM3-2, VM4-2 and VM5-2	VM1-3, VM2-3, VM3-3, VM4-3 and VM5-3	TCP-SYN flooding
4	VM1-1, VM2-1 and VM3-1	VM1-3, VM2-3 and VM3-3	UDP flood, Xpinger, HTTP flood (DoS), Fishing port scanner
5	VM1-1, VM2-1, VM3-1 and VM4-1	VM1-3, VM2-3, VM3-3 and VM4-3	UDP flood, Xpinger, HTTP flood (DoS), Fishing port scanner
6	VM1-1, VM2-1, VM3-1, VM4-1 and VM5-1	VM1-3, VM2-3, VM3-3, VM4-3 and VM5-3	UDP flood, Xpinger, HTTP flood (DoS), Fishing port scanner

Table 4. Experiments conducted in *offline* simulation of Hybrid NIDS.

	Training dataset	Testing dataset
Test-1	KDD99(10%) training dataset	KDD99 test dataset
Test-2	KDD99 (100%) training dataset	KDD99 test dataset
Test-3	CAIDA + DARPA	CAIDA + DARPA

Figure 8. Number of non-distributed intrusions detected by Hybrid-NIDS. **Figure 9.** Number of distributed intrusions detected by Hybrid-NIDS.

that Hybrid-NIDS can detect a large number of intrusions. Ideally, false positive rate should be 0. The false positive rate for individual classifiers and Hybrid-NIDS on different

datasets of intrusions is shown in figure 12. Hybrid-NIDS has a false positive rate less than 1.05%.

As shown in figure 13, true negative rate is higher (>98%) for Hybrid-NIDS. This represents that more than 98% normal records (in test dataset) are identified as normal by Hybrid-NIDS. Hybrid-NIDS has less than 2% false negatives (see figure 14). Figure 15 shows that Hybrid-NIDS achieves higher accuracy (>97%). Hybrid-NIDS takes about 19 s for inspecting 0.31 million connection profiles (see table 5).

For analysing the overall performance of Hybrid-NIDS, we have derived *weighted average* for the applied tests. *Weighted average* can be derived by the relation

$$\text{weighted_average} = \frac{\sum_{i=1}^n O_i \times NP_i}{\sum_{i=1}^n NP_i}$$

where n is number of tests, O_i is output of the i^{th} test and NP_i is number of profiles inspected in the i^{th} test.

As shown in table 6 and figure 16, Hybrid-NIDS has true positive rate >99%, false positive rate of 0.61%, true negative rate of >99%, false negative rate of 1.5% and accuracy >99%. This ensures feasibility of Hybrid-NIDS in Cloud.

In table 7, performances of recent NIDS approaches from [44–49] are compared to that of Hybrid-NIDS. These approaches use soft computing techniques for intrusion detection, and they are evaluated on KDD dataset. Hybrid-NIDS has better performance results compared with those of NIDS approaches.

Our proposed system fulfills the following requirements of Cloud IDS.

Handling high traffic volume: Hybrid-NIDS sensor (deployed on each host machine) monitors network traffic from virtual switch, and therefore, it can monitor multiple VMs. In addition, it has low detection time so that it can handle high traffic volume from VMs.

Detecting various attacks with minimum false alerts: Hybrid-NIDS detects various scanning and flooding attacks in virtual environment (e.g., port scanning, host scanning,

Figure 10. Number of alerts dropped at the central server.

Figure 11. True positive rate of Hybrid-NIDS.

Figure 13. True negative rate of Hybrid-NIDS.

Figure 12. False positive rate of Hybrid-NIDS.

Figure 14. False negative rate of Hybrid-NIDS.

TCP-SYN flooding, UDP flooding, ICMP flooding, HTTP flooding, etc.). In addition, it detects distributed attacks (from multiple VMs) by correlating alerts in the entire network. Hybrid-NIDS generates minimum false alerts (1.05%).

Fast detection: Hybrid-NIDS requires less time (19 s) for inspecting 0.31 million connections. Hybrid-NIDS uses centralized approach for correlating alerts of distributed attacks, and therefore it has communication cost $O(n)$, where n is the number of sensors sending alerts of distributed attack. In contrast with this, distributed approaches

Figure 15. Accuracy of Hybrid-NIDS.

Table 5. Detection time of Hybrid-NIDS in *offline* simulation (detection time is derived on a VM having 32 GB RAM and 2 core CPU).

	Training dataset	Testing dataset	Detection time (s)
Test-1	KDD99(10%) training dataset	KDD99 test dataset	19
Test-2	KDD99(100%) training dataset	KDD99 test dataset	19
Test-3	DARPA + CAIDA	DARPA + CAIDA	88

Table 6. *Weighted average* results of Hybrid-NIDS in offline simulation.

	True positive rate (%)	False positive rate (%)	True negative rate (%)	False negative rate (%)	Accuracy (%)
Test-1	98.04	1.04	98.96	1.96	98.22
Test-2	99.56	0.63	99.37	0.44	99.52
Test-3	99.96	0.3	99.7	0.04	99.74
Weighted average	99.28	0.61	99.39	1.42	99.33

Figure 16. Weighted average results of Hybrid-NIDS.

from [15, 16] require $O(n(n-1))$ communication cost since each sensor sends alert of distributed attack to other sensors.

Scalability: Deployment of Hybrid-NIDS on each individual host machine helps offer scalability. It captures the network traffic from virtual switch. Due to low detection time of Hybrid-NIDS, it can handle high traffic volume.

Securing each component: Hybrid-NIDS inspects VM-to-VM network traffic (for intrusion) from virtual switch, and thus it secures VMs from malicious VMs. In addition, it monitors external network traffic (through physical NIC at host machine) for intrusions, and therefore it secures host machines and VMs from external attacks.

Resistance to compromise: Hybrid-NIDS cannot be compromised by host manipulation attack since it is not deployed on individual VM. In addition, *central log* is not compromised since it is accessed only by host machines where root privilege is required.

Most of the existing Cloud IDS approaches satisfy only part of Cloud IDS requirements (viz *Handling high traffic volume*, *Detecting various attacks with minimum false alerts*, *fast detection*, *scalability*, *securing each component*, *resistance to compromise*, etc.), while Hybrid-NIDS fulfills most of the requirements with better performance results. Table 8 compares Hybrid-NIDS with the existing Cloud IDS approaches in terms of fulfilling Cloud IDS requirements.

Table 7. Performance comparison of recent NIDS approaches with Hybrid-NIDS.

Authors	Technique/ dataset	True positive rate (%)	False alert rate (%)	Accuracy (%)
Ibrahim <i>et al</i> [44]	SOM-ANN/ KDD, NSL-KDD	92.37	4.67	NR
Wang <i>et al</i> [45]	Attribute-weighted clustering KDD	95.1	5.23	NR
Long <i>et al</i> [46]	Fuzzy-SVM/ KDD	88.2	4.5	NR
Kannan <i>et al</i> [47]	Genetic algorithm + fuzzy-SVM / KDD	96.53	3.13	98.51
Hubballi <i>et al</i> [48]	BIRCH clustering/KDD	96.97	NR	97.25
Naidu and Avadhani [49]	SVM/KDD	98.03	3.65	NR
Hybrid-NIDS (proposed framework)/ (2015)	Associative, Bayesian and Decision tree/ KDD, DARPA and CAIDA	99.28	0.61	99.33

(NR—not reported)

Table 8. Comparison of Hybrid-NIDS with existing Cloud IDS approaches.

Authors	Handling high traffic volume	Detecting various attacks	Fast detection	Scalability	Securing each component	Resistance to compromise
Roschke <i>et al</i> [9]	✓	×	✓	×	P	×
Bakshi and Yogesh [17]	×	×	✓	✓	✓	✓
Mazzariello <i>et al</i> [18]	✓	×	✓	P	×	×
Lo <i>et al</i> [15]	✓	✓	×	NA	×	×
Gul and Hussain [19]	×	×	✓	×	P	×
Dastjerdi <i>et al</i> [14]	×	NA	NA	✓	×	NA
Dhage and Meshram [20]	✓	✓	×	✓	×	×
Kholidy and Baiardi [21]	✓	×	×	×	✓	✓
Gupta <i>et al</i> [8]	✓	✓	✓	×	P	×
[22]	✓	P	✓		✓	×
Our framework: Hybrid-NIDS (2015)	✓	✓	✓	✓	✓	P

(✓ – completely satisfy, P – partially satisfy, × – do not satisfy, NA – not applicable)

5. Conclusions

Cloud security concerns open up space for research to extend traditional security techniques and to investigate new techniques. Compared with traditional environment, Cloud brings additional challenges to IDS. We looked at existing Cloud IDS frameworks and identified research gaps such as detecting VM-to-VM attacks, monitoring and inspecting high volume of traffic, addressing scalability, having resistance to compromise, etc. In this research work, we proposed a security framework named as Hybrid-NIDS for Cloud. Hybrid-NIDS sensors deployed on each host machine monitor network traffic to VMs and host machines. We have applied signature-based technique followed by anomaly detection technique. Thus, anomaly detection has to analyse less number of packets, and thereby detection time is reduced there. In score calculation, weighted averaging method has helped us to improve detection accuracy. The majority voting method helped us in detecting distributed intrusions. We have validated functionality of Hybrid-NIDS using experimental Cloud-test bed. Experimental results are very encouraging. Compared with other NIDS approaches, it detects a large number of intrusions with low false alerts and high accuracy. It satisfies Cloud network security needs, viz Handling high traffic volume, Detecting various attacks with minimum false alerts, fast detection, scalability, securing each component and resistance to compromise. This proves the feasibility of Hybrid-NIDS for Cloud computing.

References

- [1] Mell P, Grance T and Gentry C 2011 The NIST definition of cloud computing (draft). *National Institute of Standards and Technology* (NIST), <http://csrc.nist.gov/publications/nist-pubs/800-145/SP800-145.pdf>
- [2] Popovic K and Hocenski Z 2010 Cloud computing security issues and challenges. In: *Proceedings of the 33rd International Convention MIPRO*, pp. 344–349
- [3] Gens F 2008 *IT Cloud services user survey, pt.2: top benefits and challenges*. International Data Corporation, <http://blogs.idc.com/ie/?p=210>
- [4] Biggs S and Vidalis S 2009 Cloud computing: the impact on digital forensic investigations. In: *Proceedings of the International Conference on Internet Technology and Secured Transactions*, pp. 1–6
- [5] Modi C, Patel D, Borisaniya B, Patel H, Patel A and Muttukrishnan R 2013 A survey on security issues and solutions at different layers of Cloud computing. *The Journal of Supercomputing* 63(2): 561–592
- [6] Modi C, Patel D, Borisaniya B, Patel A and Muttukrishnan R 2013 A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications* 36(1): 42–57
- [7] Martin L 2010 *Awareness, trust and security to shape Government Cloud adoption*. Lockheedmartin, <http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/Cloud-Computing-White-Paper.pdf>
- [8] Gupta S, Kumar P and Abraham A 2013 A profile based network intrusion detection and prevention system for securing Cloud environment. *International Journal of Distributed Sensor Networks* 9(3): 1–12
- [9] Roschke S, Feng C and Meinel C 2009 An extensible and virtualization-compatible IDS management architecture. In: *Proceedings of the Fifth International Conference on Information Assurance and Security*, pp. 130–134
- [10] Cox P 2010 *Intrusion detection in a cloud computing environment*. Techtarget, <http://searchcloudcomputing.techtarget.com/tip/Intrusion-detection-in-a-cloud-computing-environment>
- [11] Patel A, Taghavi M, Bakhtiyari K and Jnior J C 2013 An intrusion detection and prevention system in cloud computing: a systematic review. *Journal of Network and Computer Applications* 36(1): 25–41

- [12] NVD 2009 *Vulnerability summary for CVE-2009-1542*. National Vulnerability Database, <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-1542>
- [13] NVD 2007 *Vulnerability summary for CVE-2007-4993*. National Vulnerability Database, <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-4993>
- [14] Dastjerdi A V, Bakar K A and Tabatabaei S G H 2009 Distributed intrusion detection in Clouds using mobile agents. In: *Proceedings of the Third International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 175–180
- [15] Lo C C, Huang C C and Ku J 2010 A cooperative intrusion detection system framework for Cloud computing networks. In: *Proceedings of the 39th International Conference on Parallel Processing Workshops (ICPPW)*, pp. 280–284
- [16] Ram S 2012 Secure Cloud computing based on mutual intrusion detection system. *International Journal of Computer Application* 2(1): 57–67
- [17] Bakshi A and Yogesh B 2010 Securing Cloud from DDOS attacks using intrusion detection system in virtual machine. In: *Proceedings of the Second International Conference on Communication Software and Networks*, pp. 260–264
- [18] Mazzariello C, Bifulco R and Canonico R 2010 Integrating a network IDS into an open source Cloud Computing environment. In: *Proceedings of the 2010 Sixth International Conference on Information Assurance and Security (IAS)*, pp. 265–270
- [19] Gul I and Hussain M 2011 Distributed Cloud intrusion detection model. *International Journal of Advanced Science and Technology* 34: 71–82
- [20] Dhage S N and Meshram B B 2012 Intrusion detection system in Cloud Computing environment. *International Journal of Cloud Computing* 1(2): 261–282
- [21] Kholidy H A and Baiardi F 2012 CIDS: a framework for intrusion detection in Cloud systems. In: *Proceedings of the 2012 Ninth International Conference on Information Technology: New Generations (ITNG)*, pp. 379–385
- [22] Idrees F, Rajarajan M and Memon A Y 2013 Framework for distributed and self-healing hybrid intrusion detection and prevention system. In: *Proceedings of the International Conference on ICT Convergence (ICTC)*, pp. 277–282
- [23] Snort 2014 *Snort-home page*. Snort Tool, <https://www.snort.org/>
- [24] Modi C N, Patel D R, Patel A and Rajarajan M 2012 Integrating Signature Apriori based Network Intrusion Detection System (NIDS) in Cloud Computing. In: *Proceedings of the 2nd International Conference on Communication, Computing & Security [ICCCS-2012]*, pp. 905–912
- [25] Modi C N, Patel D R, Patel A and Rajarajan M 2012 Bayesian classifier and Snort based network intrusion detection system in cloud computing. In: *Proceedings of the 2012 Third International Conference on Computing Communication Networking Technologies (ICCCNT)*, pp. 1–7
- [26] Han J 2005 *Data mining: concepts and techniques*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA
- [27] Modi C, Patel, D, Borisaniya B, Patel A and Muttukrishnan R 2012 A novel framework for intrusion detection in Cloud. In: *Proceedings of the Fifth International Conference on Security of Information and Networks*, pp. 67–74
- [28] NSL-KDD Data Set 2014 NSL-KDD intrusion dataset. NSL-KDD, <http://nsl.cs.unb.ca/NSL-KDD/>
- [29] Carstens T 2014 *The Sniffer's guide to raw traffic*. A libpcap tutorial, <http://yuba.stanford.edu/~casado/pcap/section1.html>
- [30] KDD Cup Data 1999 *KDD intrusion dataset*. KDD, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [31] Chan A P F, Ng W W Y, Yeung D S and Tsang E C C 2005 Comparison of different fusion approaches for network intrusion detection using ensemble of RBFNN. In: *Proceedings of the 2005 International Conference on Machine Learning and Cybernetics*, pp. 3846–3851
- [32] Oza N C and Tumer K 2008 Classifier ensembles: select real-world applications. *Information Fusion* 9(1): 4–20
- [33] Modi C N and Patel D 2013 A novel hybrid-network intrusion detection system (H-NIDS) in cloud computing. In: *Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pp. 23–30
- [34] Bidakhvidi M A 2015 *Net tools*. Net Tools, <http://users.tele.net.be/ahmadi/nettools.htm>
- [35] Pachghare S V 2011 *SYN flooding using SCAPY and prevention using iptables*. Linux forum, <http://www.linuxforum.com/2011/10/syn-flooding-using-scapy-and-prevention-using-iptables/>
- [36] Moon S 2009 *SYN flood DOS attack with C source code (Linux)*. Binary tides, <http://www.binarytides.com/syn-flood-dos-attack/>
- [37] Nmap 2014 *Nmap—the network mapper*. Nmap, <http://nmap.org/>
- [38] Arora H 2011 *C socket programming for Linux with a server and client example code*. The Geek Stuff, <http://www.thegeekstuff.com/2011/12/c-socket-programming/>
- [39] DARPA dataset 1999 *DARPA intrusion detection data sets*. DARPA, <http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/>
- [40] Hick P 2007 *The CAIDA DDoS attack 2007 dataset*. CAIDA Datasets, <https://data.caida.org/datasets/security/ddos-20070804/>
- [41] Sathya S S, Ramani R G and Sivaselvi K 2011 Discriminant analysis based feature selection in KDD intrusion dataset. *International Journal of Computer Applications* 31(11): 1–7
- [42] Singh S and Silakari S 2009 An ensemble approach for feature selection of Cyber Attack Dataset. *International Journal of Computer Science and Information Security* 6(2): 297–302
- [43] Kayacik N and Heywood M 2005 Selecting features for intrusion detection: a feature relevance analysis on KDD 99 intrusion detection datasets. In: *Proceedings of the The 3rd Annual Conference on Privacy, Security and Trust (PST)*
- [44] Ibrahim L M, Basheer D T and Mahmod M S 2013 A comparison study for intrusion database (KDD99, NSL-KDD) based on self organization map (SOM) artificial neural network. *Journal of Engineering Science and Technology* 8(1): 107–119
- [45] Wang L, Zhang S, Li Y, Wu R and Yu Y 2013 An attribute-weighted clustering intrusion detection method. *Journal of Engineering Science and Technology* 8(6): 1278–1284
- [46] Long Y, Ouyang J and Sun X 2013 Network intrusion detection model based on fuzzy support vector machine. *Journal of Networks* 8(6): 1387–1394
- [47] Kannan A, Maguire G Q, Sharma A and Schoo P 2012 Genetic algorithm based feature selection algorithm for effective intrusion detection in Cloud networks. In:

- Proceedings of the International Conference on Data Mining*, pp. 416–423
- [48] Hubballi N, Biswas S and Nandi S 2013 Towards reducing false alarms in network intrusion detection systems with data summarization technique. *Security and Communication Networks* 6(3): 275–285
- [49] Naidu R C A and Avadhani P S 2012 A comparison of data mining techniques for intrusion detection. In: *Proceedings of the 2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, vol. 6(3), pp. 41–44