Paper

Detection of DDoS Attacks in OpenStack-based Private Cloud Using Apache Spark

Shweta Gumaste, Narayan D. G., Sumedha Shinde, and Amit K

KLE Technological University, Hubballi, India

https://doi.org/10.26636/jtit.2020.146120

Abstract—Security is a critical concern for cloud service providers. Distributed denial of service (DDoS) attacks are the most frequent of all cloud security threats, and the consequences of damage caused by DDoS are very serious. Thus, the design of an efficient DDoS detection system plays an important role in monitoring suspicious activity in the cloud. Real-time detection mechanisms operating in cloud environments and relying on machine learning algorithms and distributed processing are an important research issue. In this work, we propose a real-time detection of DDoS attacks using machine learning classifiers on a distributed processing platform. We evaluate the DDoS detection mechanism in an OpenStack-based cloud testbed using the Apache Spark framework. We compare the classification performance using benchmark and real-time cloud datasets. Results of the experiments reveal that the random forest method offers better classifier accuracy. Furthermore, we demonstrate the effectiveness of the proposed distributed approach in terms of training and detection time.

Keywords—cloud, DDoS, distributed processing, OpenStack, Apache Spark, random forest.

1. Introduction

Cloud computing is a model of computing resources, such as storage devices, servers, services, applications and networks, that are accessible from any location via the Internet. Cloud technology enables hardware infrastructure to be considered a common shared service on which applications and different services are provided in a cost-effective manner.

Private cloud is one of the deployment models which provides computing services offered over a private internal network. However, security issues are the primary concern affecting the data entered by and the cloud services relied upon by users. DDoS attacks are one of the major threats that are faced in cloud computing [1], [2]. The main strategy behind a DDoS attack is to target the victim by submitting a lot of requests in a distributed manner in order to exhaust the victim's resources, and therefore rendering the target resources unavailable, over a specific period of time, to the legitimate users. Hence, detection of such attacks in the cloud requires that special attention be paid. OpenStack is an open-source software platform that provides a set of software tools for creating and managing virtual servers. When a private cloud is deployed using OpenStack, the firewall alone is not sufficient to counter DDoS attacks, so the administrator requires an additional intrusion detection system supplementing the firewall in order to detect DDoS attacks.

Classification algorithms are utilized to classify traffic packets and to predict intrusions. Selection of the appropriate machine learning algorithm for each dataset is an important issue. For accurate classification of data, one must select the right classifier. Thus, the estimation of the algorithms and the comparison of their performance are necessary while choosing the classifier. Furthermore, there is a trade-off between choosing classifiers based on their accuracy, precision and computational time. Most classifiers suffer from issues in terms of the pace of intrusion detection and the related performance. This motivates researchers to seek new approaches enabling more efficient and faster detection of DDoS attacks.

The system we propose follows an anomaly-based detection approach in a private cloud. Network traffic from the cloud is captured and analyzed using a model trained to detect intrusions. Next, we deploy these algorithms on a distributed platform for faster processing. The Apache Spark framework is used for the detection of anomalous traffic in the data captured. Spark uses a stream-oriented approach, relying on in-memory computations to process real-time data faster. This helps in real-time detection of DDoS attacks. An efficient classification model is selected for the classification of traffic packets.

The contributions of the work are as follows:

- we designed Spark as a service in an OpenStack cloud for provisioning of on-demand Spark clusters,
- we determined the best classifier for detection of DDoS attacks,
- we designed a DDoS attack detection system in a distributed framework using Spark,

- we evaluated the proposed system using a real-time cloud testbed.

The rest of this paper is organized as follows. In Section 2, we discuss DDoS attacks, related works and the Apache Spark framework. Section 3 describes the proposed methodology and the algorithms used. Section 4 presents the results of machine learning classifiers and the DDoS detection system deployed on the experimental testbed. Finally, conclusions and future work are presented in Section 5.

2. Background Study

In this section, we introduce the concept of DDoS attacks. Next, we discuss the related research on DDoS attacks detection, focusing on cloud environment and distributed processing. Furthermore, we discuss features of Apache Spark in order to facilitate understanding of the distributed approach to make the system scalable and efficient.

2.1. Distributed Denial of Service Attacks

DDoS attacks are the most widely used type of cyber-attacks where the attacker uses a botnet consisting of malware-infected machines. As shown in Fig. 1, the attacker uses a command and control center and the botnet to perform a DDoS attack on the victim machine. The foe machine specifies the attack type and the victim's IP address to the command and control server. Then, the command and control server sends this information to the botnet network. Multiple machines comprising the botnet network send a stream of attack requests to the victim machine, by throwing a wrong input or by sending a huge number of requests simultaneously. Even after the victim server responds to this request, the botnet machines throw the same request again and again. Thus, resources of the victim server become exhausted, the server crashes and its performance degrades. The scenario in which the victim's server is bombarded with a huge number of requests leading to server crash, is called a denial of service, as authorized

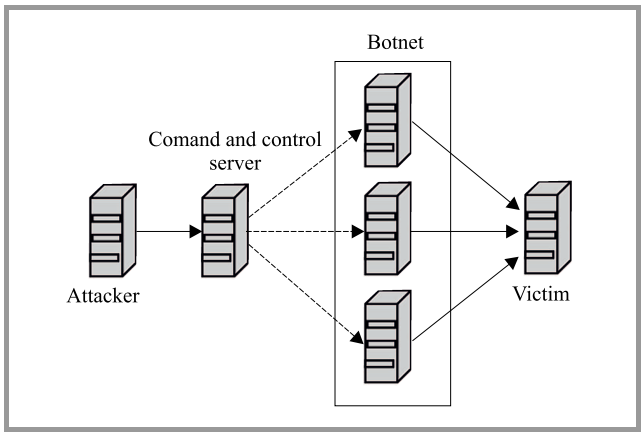

Fig. 1. DDoS attack scheme.

users are unable to access the server or a given service. As the attacks originate from different machines of the botnet, they are referred to as distributed DoS attacks. As the number of DDoS attacks has been increasing in recent years, detection or mitigation of DDoS attack has become an important issue.

2.2. Related Work

An intrusion detection system (IDS) is a security mechanism deployed for detecting malicious activities. IDS involves two types of detection methods that may be signature- or anomaly-based. Signature-based IDS uses the signature of known attacks. Snort is an open source IDS that relies on signature-based techniques for detecting attacks know from experiments [3]. The authors presented a DDoS detection mechanism based on machine learning and signature detection techniques. In [4], the authors proposed an NIDS framework operating in the cloud and consisting of a snort and signature-based apriori algorithm to detect known attacks in the network. However, signature-based mechanisms cannot detect unknown attacks or variants of known attacks.

Anomaly-based detection captures network data or logs to detects anomalies. It detects a new form of attacks rather than their signatures only. In [5], the authors proposed an entropy- based DDoS attack detection approach deployed in a cloud-based network system. The normalized entropy value is calculated to check changes in the randomness between packet header field samples. This system is maintained and represented by a third party which sends a notification to the client. In [6], the authors proposed a DDoS detection mechanism that focuses on various features of attack packets. The features are obtained by studying incoming network traffic. The features are then analyzed using radial basis function (RBF) neural networks. The authors evaluated the proposed method using a simulated network and the UCLA dataset. In [7], the authors presented an IDS operating in the cloud environment and focusing on the software-as-a-service (SaaS) model to provide application level security to the cloud customer. They consider a virtual private network for information exchange. However, research has not been conducted with regard to real cloud infrastructures. The authors in [8] present a DDoS attack detection methods using OpenStack cloud. OpenStack is an open source cloud operating system which used for academic research and for commercial cloud applications.

The above techniques do not use distributed processing for detecting DDoS attacks. In cloud environments, huge numbers of packets or connections with customers' networks exist. Thus, protecting the organizations' networks from DDoS attacks requires huge amounts of processing power. So, there is a trend towards using distributed environments, like Hadoop and Spark, for detection of DDoS attacks. The work proposed by [9] focuses on a reliable DDoS attack detection approach based on HTTP GET flooding. It uses MapReduce processing for flooding detection. The results

show that processing time is better than in snort detection. In [10], the authors designed a DDoS prevention scheme based on the Hadoop framework, using the MapReduce technique and a genetic algorithm (GA) to detect and prevent DDoS attacks. They generate the filtering rules with an entropy-based detection scheme. The rules are updated accordingly as well. In [11], the authors presented a DDoS attack detection system utilizing classification algorithms in the Spark framework. In this work, T-shark is used for network analysis. The fuzzy logic algorithm chooses the next candidate for classification. Classification algorithms are designed to predict DDoS attacks affecting traffic packets.

The authors of [12] proposed a DDoS detection system which uses the feature selection technique known as ensemble-based multi-filter to select optimal features by combining the output of four filter methods. The method is evaluated using an NSL-KDD dataset and a decision tree classifier. However, the discussion covers the efficient feature selection technique only. Furthermore, the system suffers from an overhead in the form of a processing delay. In [13], the authors use the Hadoop map to reduce the architecture for faster processing of log files. Furthermore, abnormal behavior of sources that generate packets erratically is predicted. The architecture uses prediction-based time series analysis, which helps in faster detection of suspicious hosts. However, research is limited to threshold-based detection.

In [14], the authors proposed a live DDoS detection mechanism to detect 4 types of flooding attacks in real-time, using the map reduce approach. The results were validated on a real-time testbed using low cost hardware. The authors of [15] proposed a technique to detect DDoS attacks using the Hadoop framework for processing and analyzing large scale attacks in real-time. The entropy of source addresses is used as a metric for detection. The results are validated using the testbed. In [16], the authors proposed a distributed and collaborative approach to the processing of a large amount of data by distributing it among a number of mappers and reducers. The proposed work is validated using benchmark datasets, as well as real datasets generated using the experimental testbed, based on various metrics.

In [17], the authors proposed a distributed and collaborative technique for early detection of DDoS attacks and flash crowds. However, due to non-cooperation between various ISPs, it is difficult to modify routers to detect attacks in real-time. The authors validate the results using a real-time testbed. In [18] the authors designed a DDOS attack detection mechanism using different classifiers for OpenStack cloud. The authors validate the results using a testbed with real-time and benchmark datasets. The authors also proposed a DDoS attack detection mechanism in the SDN environment using a Mininet simulator [19]. The authors use SVM and DNN to classify attacks by capturing real-time packets. However, the authors of [18], [19] have not used any distributed environments for the detection of DDoS attacks. As the millions of packets are captured per second in the cloud, the design of a distributed processing approach relied upon by the detection system constitutes a crucial research issue.

2.3. Distributed Processing using Spark

Apache Spark is a distributed platform built on the YARN infrastructure. Spark supports in-memory processing which is iterative in nature. Spark supports many languages, such as Scala, R, Java and Python. As Python is an efficient high-level language and allows us to develop the system faster, we use Python. The proposed system employs RDD, Spark streaming and MLlib features. RDD is the key for Spark to offer failure recovery and data dependency functionalities. RDD stores intermediate results in the memory, which significantly improves computation speed. Unlike MapReduce in Hadoop, MapReduce in Spark is well packaged into RDD.
MLlib is Spark's distributed machine learning (ML) library. It provides ML algorithms with libraries used for such purposes as regression, classification, clustering, collaborative filtering, etc. Spark streaming is used for real-time calculation that relies on RDD. RDD helps in seamless connection with Spark to fuse historical data and real-time data. Spark streaming divides the streaming data into small time intervals and into RDD data sets. Later, it processes the RDD in batches and helps process complex streaming data faster. Thus, RDD helps in processing huge numbers of packets in real-time, extracts the features and detects the attacks affecting cloud environments.

3. Proposed Methodology

This section offers a description of the proposed system that allows to detect DDOS attacks using the distributed processing approach. The chapter has been divided into two subsections, focusing on the design of a Spark cluster as a service on an OpenStack private cloud, and on the detection of DDoS attacks using real-time cloud traffic and the KDD Cup dataset, respectively.

3.1. Spark as a Service

The design of this service involves setting up an OpenStack private cloud and configuring a multi-node setup. The multi-node setup is done on virtual machines (VMs) by configuring compute, neutron and controller nodes. The VMs are spawned on the compute node of OpenStack. Deployment of the Spark cluster is done, firstly, by executing the machines with single node Spark configuration steps. One of the single nodes is designated as a master and the remaining nodes are configured to be working as slaves under the supervision of the master. The user interface is provided by relying on the Django framework with Python. The user can select the number of slaves required. The user is provided with options to select the number of VCPUs, while RAM and disk size are required for the master

and the slaves based on standard templates. The user can also create his/her own flavors for slaves. Once the user clicks the launch button, the Spark cluster will be installed in the cloud with the help of an automation script executed at the OpenStack backend using controller and compute nodes. The design includes a connection between Spark and OpenStack, as shown in Fig. 2.

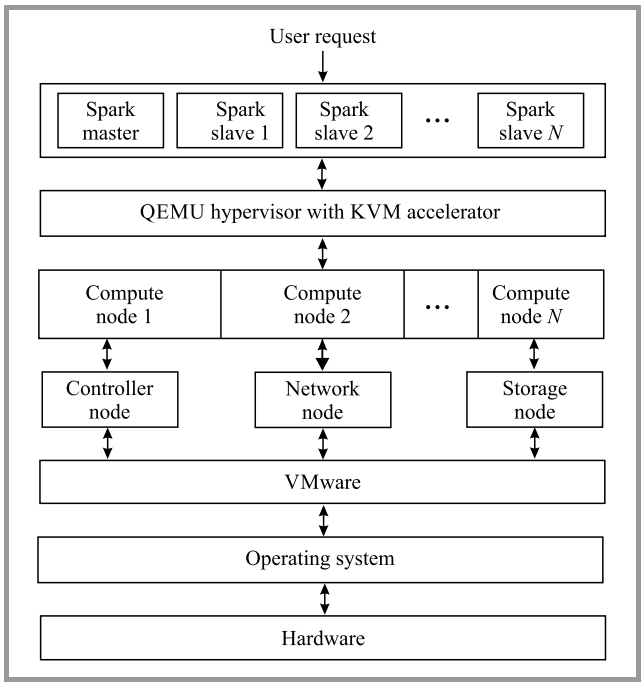

Fig. 2. Spark as a service.

The algorithm for automation of Spark as a service is described in Algorithm 1. Spark is built using Hadoop architecture. Hence, it involves Hadoop installation. Hadoop uses a Java framework. For big data processing MapReduce is required, which is written in Java. Java is used for analyzing and processing large data sets. So, JDK is installed for supporting the Hadoop platform.

3.2. DDoS Detection System

The incoming packets get captured through a packet sniffer stored in the local disk as packet log files in CSV format. In real life, these log files need faster processing. So, a distributed computing platform is used for big data analysis and for faster detection of attacks. Anomaly-based IDS treats any deviation from the normal pattern as an attack. As shown in Fig. 3, the model of the proposed system consists mainly of four modules, i.e. packet sniffer, pre-processing using Spark steaming, classification, and detection. All of them are discussed in detail below.

3.3. Packet Sniffer

Packet sniffer is designed using socket programming to capture and analyze network traffic by connecting two virtual machines on a cloud network in order to enable them to

Algorithm 1. Spark as a service on OpenStack cloud

Input: MapReduce cluster name C, number of slave nodes N, configuration of master and slave nodes S, keypair name K

Output: Cluster with required number of nodes

1: Create VMs based on master and slave size S using OpenStack SDK
2: Create a keypair K
3: Create one node as master node
4: Create N slave nodes with configuration S
5: Assign IP address for the master node
6: Change the hostname in hosts file and hostname file
7: **For** N node
8: Assign the IP address to slave nodes
9: Change the host name in hosts and hostname files
10: **End for**
11: Add all the IPs of master and slave nodes in private IP file
12: Install the services on master and slave nodes
13: Configure SSH on master node
14: **For** N nodes
15: Configure SSH on slave node
16: **End for**
17: Send the keypair from master to slave nodes
18: Disable the password authentication
19: Do the Hadoop name node format on master node
20: Start all the services on master node
21: Write all the IPs to one file
22: Mail the keypair and master node IP address

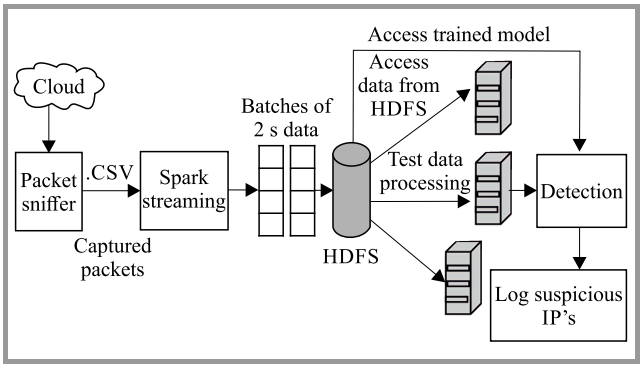

Fig. 3. Proposed DDoS attack detection system using Spark.

communicate with each other. Packet sniffer collects network traffic and identifies information pertaining to each packet. This raw data (packets) is sent to the Spark processor for feature extraction, as shown in Fig. 3.

Algorithm 2 is designed for capturing network traffic in the Neutron node of the OpenStack cloud. Initially, the socket library is imported and the creation of a raw socket follows. It listens on a particular port of the neutron node's IP, while the other socket reaches out to other VMs to form a connection. The listener socket is formed at the server while the client reaches out to the server. Packet sniffer gathers raw binary data from the connection established. The selected network interface is switched to promiscuous

Algorithm 2. Capturing of packets using packet sniffer

Input: Neutron node IP interface
Output: CSV file

1: Create raw socket
2: Bind a raw socket to Eth0 interface of neutron node
3: Set the interface to promiscuous mode
4: Capture data link layer (Ethernet) header
5: Retrieve IP/ICMP/TCP header fields by parsing Ethernet frame
6: Extract the features as per Table 1
7: Write the extracted features to CSV file

mode. Promiscuous mode is used to direct all network traffic through the chosen network interface. It captures various fields of the chosen protocol headers.
The packet sniffer module is designed to collect the captured network data and to extract packet information by verifying its protocol. The packet sniffer module captures incoming packets for a required period of time. All information pertaining to the protocol headers is stored as a CSV file. A DDoS attack is simulated in the cloud environment with a view to sending packets with large amounts of data, and to from multiple connections to simulate connection flooding and to captures the attack. The features captured in this process are shown in Table 1.

Table 1
Dataset features captured

Features	
Source MAC	Acknowledgement number
Destination MAC	SYN bit
Protocol	ACK bit
Service	FIN bit
Source address	Src bytes
Destination address	Time stamp
Sequence number	

3.4. Processing of Captured Data

Real-time data captured in the previous module needs to be processed. The processing of a dataset is performed to obtain the specific features required for testing the model. The distributed computing approach is introduced to accelerate the process. Spark is one of the distributed frameworks that enable processing based on in-memory computing. Initially, the features are recorded once every millisecond. Additionally, the derived features need to be computed for the past two seconds of the connection. The data stream captured by the sniffer module is pre-processed in a batch of 2 s. The algorithm is designed to perform certain actions for feature selection, as not all features that are captured contribute to DDoS detection. Thus, it is important to select the required features. The pre-processor sends this information to the detection engine. Later, the dataset is processed in a big data environment using the Spark engine.

Algorithm 3. Processing of the captured data

Input: Generated log file CSV
Output: Processed test dataset

1: Count = number of connections to the same host
2: Src_bytes = number of data bytes from source to destination
3: Dest_bytes = number of data bytes from destination to source
4: Srv_count = number of connections to the same service
5: Same_srv_rate = percentage of connections to the same service
6: Initialize Spark context and Spark streaming context
7: Stream captured data every 2 s
8: Read the captured data as dstream
9: **For** every RDD from dstream
10: Convert RDD to dataframe
11: Initialize Count, Src_bytes, Dest_bytes, Srv_count, Same_Srv_rate to zero
//Apply map and reduce to get the features
12: **For** each connection i.e. combination of protocol type and service
13: Apply map and reduce technique to compute the Count, Src_bytes, Dest_bytes, Srv_count, Same_Srv_rate
14: **End for**
15: Append all computed features along with protocol type and service to data frame
16: Write transformed dataframe to a file
17: **End for**

The processing of captured data collected through the packet sniffer is represented in Algorithm 3. This is basically performed for the test dataset collected and the results are used for further detection analysis. Distributed processing is carried out on the Spark framework. The packets sniffed over a specific duration serve as an input for this module. The processing is performed for each 2 s intervals of data in order to analyze network traffic. Initially, Spark context and configuration are created. The packet log data file is read and converted as a dstream object. Finally, each RDD of dstream is converted into a Spark data frame and the selected features are computed using the map reduce function (Protocol_type, Service, Src_bytes, Dest_bytes, Count, Srv_count, Same_srv_rate).

3.5. Classification using ML Algorithms

Machine learning is an effective technique allowing to detect any anomaly-based attacks. Real-time detection of a DDoS attack in a cloud environment requires a trained model. This module builds a trained model using a real-time dataset. The real-time training dataset is generated by recording network activity over the period of 24 hours by simulating an ICMP flooding attack. We compare three

classification algorithms based on accuracy, precision and false-positive rate parameters and choose the best classifier for our model. Such classification algorithms as random forest, decision tree and logistic regression are used for modeling the training dataset. We apply these classification algorithms to both the benchmark dataset and the real-time dataset. Later, we use the Spark MLlib library to implement the selected classification algorithm. As the Spark framework is designed to work in a distributed environment, we use a distributed file system. The selected training model is stored in HDFS for further retrieval. This trained model is accessed by the detection module to detect attacks as shown in Fig. 3.

3.6. Detection Module

The process of identifying anomalous attacks in normal traffic is known as detection. Existing cloud behaviors related to attacks are modeled using the trained model. As shown in Algorithm 4, a real-time test dataset is generated using the packet sniffer, by simulating a DDoS attack. This test dataset is processed using Spark streaming. The captured data is considered as a test dataset and is fed into the trained classifier. If a connection is considered to constitute an attack, then we log the suspicious IPs. The administrator checks the suspicious IP addresses and can observe the traffic anomaly over a period of time and may block the IP addresses. All four modules are implemented on the private cloud testbed.

Algorithm 4. Detection module

Input: Processed test dataset
Output: List of suspicious IP's

1: Capture the network packets for training using packet sniffer
2: Compute the features using Spark streaming
3: Read processed test dataset
4: Feed the processed test dataset into trained model to check for intrusion
5: **If** intrusion is detected **Then**
6: Log the suspicious IP's for admin analysis
7: **End if**

4. Results and Discussions

In this section, we describe the configuration of the experimental OpenStack-based cloud testbed. We also discuss the results obtained while selecting the machine learning classification algorithm. Finally, we discuss the training and prediction times for the selected ML classifier, execution time for detection of multiple slave nodes, and duration for capturing various packets.

4.1. OpenStack Cloud Environment

OpenStack [20], is a real-time cloud environment used to capture network data and to classify attacks. The private cloud setup comprises 5 nodes, namely a controller node, a neutron node and three compute nodes. The controller node provides OpenStack services and APIs to carry out the various tasks performed by the network and by the compute nodes. The neutron node helps in networking services, such as DHCP which assigns IPs to virtual machine instances. The compute node provides VM nodes using a hypervisor. Details of the real-time private cloud setup are described in Table 2.

Table 2
OpenStack testbed configuration

Nodes	IP Address	Memory
Controller	192.168.31.2	1 GB
Neutron	192.168.31.3	1 GB
Compute node 1	192.168.31.4	4 GB
Compute node 2	192.168.31.5	4 GB
Compute node 3	192.168.31.6	4 GB

The packet sniffer module was designed for capturing network traffic. The objective of the analysis is to understand the behavior experience during an attack behavior on the private cloud testbed.

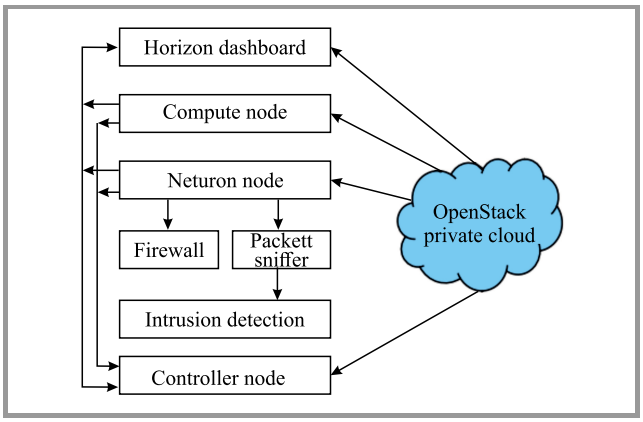

Fig. 4. Integration of packet sniffer and IDS on OpenStack.

Figure 4 shows the integration of the packet sniffer and IDS in an OpenStack cloud. The packet sniffer module is installed on the neutron node of the cloud to capture activities of the cloud. An ICMP flooding attack is performed to simulate attacks on the cloud. The behavior of the cloud is captured. The intrusion detection system designed on the spark cluster is deployed in the cloud to detect any suspicious activity.

4.1.1. Apache Spark Platform

In this experiment, we have used one master node and varying numbers of slave nodes (up to 3). We used Apache Spark [21] for distributed processing. Spark clusters are generated with the use of OpenStack virtual machines, relying on the automated process described earlier. It should

be noted that the configuration in both the slave machines and the master machine are the same.
The Spark configuration has been set as follows:

- Spark executor instances = 4,
- Spark executor cores = 10,
- size of Spark executor memory = 20,
- size of Spark driver memory = 9 B.

As the Spark framework is designed to work in a distributed environment, the Hadoop distributed file system (i.e. HDFS) was adopted.

4.2. Selection of Machine Learning Classifier

We use three parameters to select the best machine learning classifier:

Accuracy. Accuracy is the percentage ratio between the number of correctly classified samples from the dataset and the total number of samples of the dataset, and it reflects the discrimination capability of the classifier:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \cdot 100\% \ ,$$

where TP – true positive, TN – true negative, FP – false positive, FN – false negative.

False Positive Ratio (FPR). FPR indicates the proportion between wrong classified normal samples and all normal samples. FPR is a parameter that indicates the probability of falsely rejecting the null hypothesis for a particular test:

$$FPR = \frac{FP}{FP+TN} \ .$$

Precision. Precision is defined as the number of true positives divided by the number of true positives with the addition of the number of false positives. Precision describes the ability of a classification model to return relevant instances only:

$$Precision = \frac{TP}{FP+TP} \cdot 100$$

Classification is a major factor in identifying an intrusion attack. It is essential to deploy the best data model to obtain accurate classification results. Table 3 presents a comparison of performance of 3 ML classifiers, namely random forest, decision tree, and logistic regression. Furthermore, accuracy, false positive rate and precision are defined for two types of datasets. The real time dataset is used as training dataset and cross validation is applied. Accuracy is noted for all 3 algorithms, i.e. random forest, decision tree and logistic regression. For KDD Cup 99 dataset the same pattern of accuracy is observed, but the accuracy level being higher than in the case of the real time dataset. From Fig. 5, one may observe that the random forest model offers better accuracy and precision, which renders it suitable for detection.

Table 3
Comparison of three classification models

Dataset	ML model	Accuracy	FPR	Precision
KDD Cup	Random forest	99.21%	0.003	99.91%
	Decision tree	98.82%	0.024	99.37%
	Logistic regression	84.97%	0.095	98.84%
Real time	Random forest	94.40%	0.111	89.87%
	Decision tree	88.24%	0.208	78.74%
	Logistic regression	81.43%	0.293	72.42%

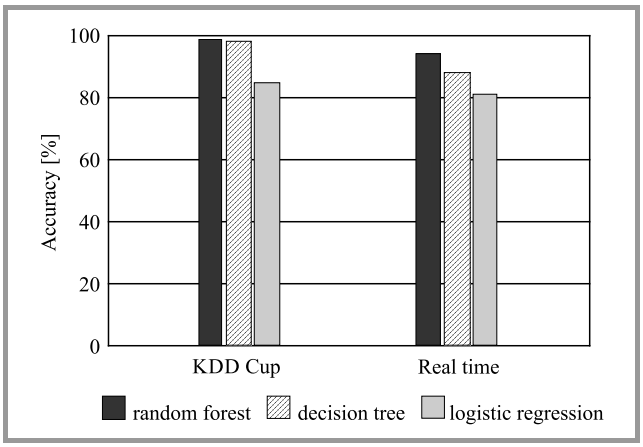

Fig. 5. Accuracy of ML models for two datasets.

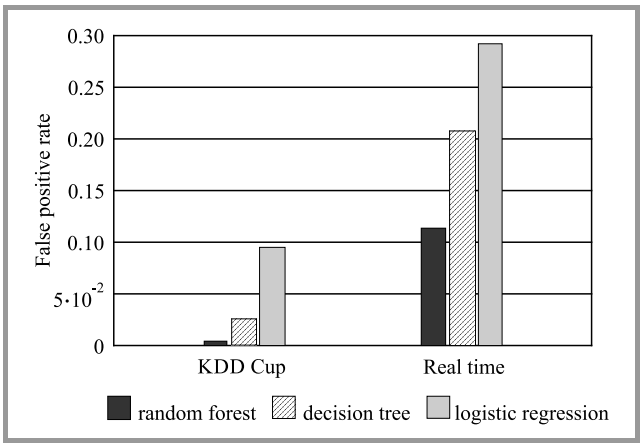

Fig. 6. False positive rate of ML models for two datasets.

The false positive ratio (FPR) is the most important parameter in anomaly detection. As the traffic pattern in the cloud network changes along with the varying intervals and events, traffic bursts are experienced. Thus, a sudden change in traffic volume may result in more FPR. This may trigger a lot of alerts for the administrator. Thus, choosing a classifier with better FPR is important in DDoS attack detection. As shown in Fig. 6, the value of FPR for the KDD Cup 99 dataset is 0.003 for random forest, 0.022 for decision tree and 0.095 for logistic regression algorithms, respectively. The value of FPR for the real time dataset is 0.111 for random forest, 0.208 for decision tree and 0.293

for logistic regression. The rate of detection of false positive samples from benchmark and real time datasets is, in random forest, lower than in the case of logistic regression and decision tree algorithms. In summary, the random forest algorithm performs better in terms of accuracy, FPR, and precision. Thus, we choose this classifier for detection of DDoS attacks in a distributed environment.

4.3. Results using Apache Spark

In this section, we discuss the results obtained using the random forest model, with different parameters and scenarios applied.

The process of providing the random forest model with training data to learn is known as training time. The learning algorithm finds patterns in the training data that map the input data attributes to the target, and it outputs an ML model that captures these patterns. The time taken for this process is training time. Figure 7 provides different training times along different spark cluster sizes for the random forest algorithm. MLlib libraries of Spark provide training and prediction times while training the classifier. The results illustrate processing speed increases as cluster size increases.

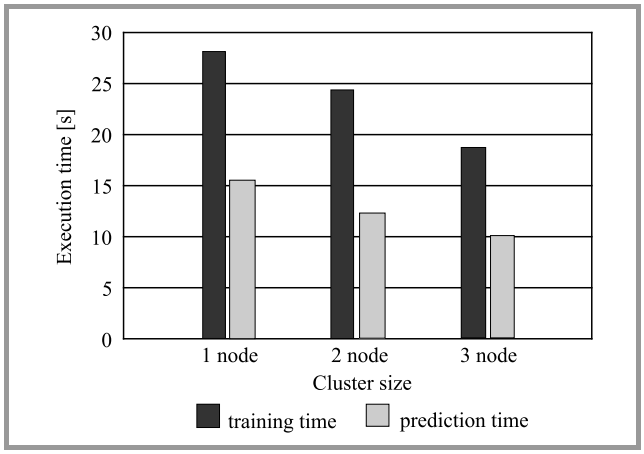

Fig. 7. Training and prediction time vs. different cluster sizes.

Figure 8 shows that as the Spark cluster size increases, the DDoS attack detection time reduces for distributed processing using Spark. The packet sniffer is executed for 4000 s and the CSV file is created and used as an input dataset for the detection of DDoS attacks. The Spark distributed detection algorithm is executed for the random forest classification model and the captured test dataset. The figure shows also a comparison between non-distributed processing and distributed processing using Spark for the scenario in question. The distributed detection process varies with different cluster sizes, such as a single node, 2 node and 3 node master-slave Spark cluster, as shown in Fig. 8.

The detection time is executed in a 2-node Spark cluster for different durations of the packets captured. The packet sniffer is executed for 10, 20 and 30 seconds ·1024 and log CSV files are created. These log files are tested against

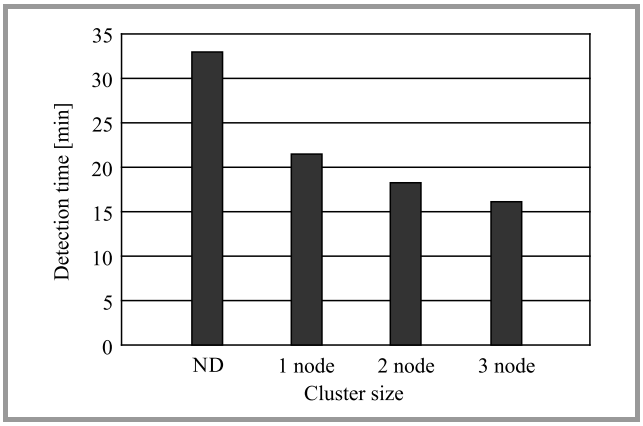

Fig. 8. Detection time vs. different cluster sizes.

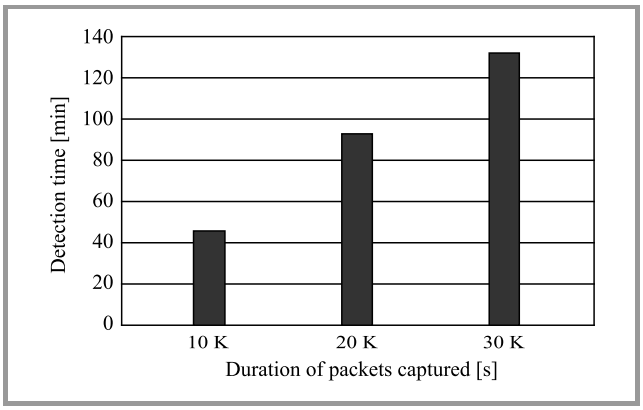

Fig. 9. Detection time vs. duration of captured packets.

the classification model to detect the attacks. The detection time is longer for the larger duration of captured packets. Based on the Fig. 9, we infer that as the capturing time increases, log file size increases. Thus, the execution time increases.

5. Conclusions

This work proposed a system for efficient detection of DDoS attacks in a private cloud environment. Initially, we designed Spark as a service in an OpenStack-based private cloud for on- demand provisioning of Spark clusters. Then, we evaluated three machine learning models using benchmark and real-time datasets and selected the random forest model for the classification of DDoS attacks. Then, we used this algorithm in the distributed environment. By using RDD Spark streaming, we performed the pre-processing, training and testing stages. The results reveal that the time for detecting DDoS attacks is reduced and the detection efficiency is improved significantly thanks to the advantages offered by the Spark framework. OpenStack was used to setup the private cloud with which the intrusion detection system is integrated.

As future work, we plan to use a deep learning model and detect different types of DDoS attacks. Furthermore,

the work may be extended to include parameter tuning of the Spark framework for increased computational efficiency.

References

[1] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments", *IEEE Access*, vol. 7, pp. 80813–80828, 2019 (DOI: 10.1109/ACCESS.2019.2922196).

[2] N. Agrawal and S. Tapaswi, "Defense mechanisms against DDoS attacks in a cloud computing environment: state-of-the-art and research challenges", *IEEE Commun. Surv. & Tutor.*, vol. 21, no. 4, pp. 3769–3795, 2019 (DOI: 10.1109/COMST.2019.2934468).

[3] M. Zekri, S. E. Kafhali, N. Aboutabit, and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments", in *Proc. 3rd Int. Conf. of Cloud Comput. Technol. and Appl. CloudTech 2017*, Rabat, Morocco, 2017, pp. 1–7 (DOI: 10.1109/CloudTech.2017.8284731).

[4] C. N. Modi, D. R. Patel, A. Patel, and M. Rajarajan, "Integrating signature apriori based network intrusion detection system (NIDS) in cloud computing", *Procedia Technol.*, vol. 6, pp. 905–912, 2012 (DOI: 10.1016/j.protcy.2012.10.110).

[5] A. S. Syed Navaz, V. Sangeetha, and C. Prabhadevi, "Entropy based anomaly detection system to prevent DDoS attacks in cloud", *Int. J. of Comp. Appl.*, vol. 42, no. 15, pp. 42–47, 2013 [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1308/1308.6745.pdf

[6] R. Karimazad and A. Faraahi, "An anomaly-based method for DDoS attacks detection using RBF neural networks", in *Proc. of Int. Conf. on Netw. and Electron. Engin. IPCSIT 2011*, vol. 11 [Online]. Available: http://ipcsit.com/vol11/9-ICNEE2011-N019.pdf

[7] A. Zarrabi and A. Zarrabi, "Internet intrusion detection system service in a cloud", *Int. J. of Comp. Sci. Issues*, vol. 9, iss. 5, no. 2, pp. 308–315, 2012 [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=1F3769647FE79FC6E52BFD98BF7C503C?doi=10.1.1.401.7096&rep=rep1&type=pdf

[8] H. Gajjar and Z. Malek, "A survey of intrusion detection system (IDS) using OpenStack private cloud", in *Proc. 4th World Conf. on Smart Trends in Syst., Secur. and Sustainab. WorldS4 2020*, London, United Kingdom, 2020, pp. 162–168 (DOI: 10.1109/WorldS450073.2020.9210313).

[9] J. Choi, C. Choi, B. Ko, D. Choi, and P. Kim, "Detecting Web based DDoS attack using MapReduce operations in cloud computing environment", *J. of Internet Serv. and Inform. Secur.*, vol. 3, no. 3/4, pp. 28–37, 2013 [Online]. Available: http://isyou.info/jisis/vol3/no34/jisis-2013-vol3-no34-03.pdf

[10] M. Mizukoshi and M. Munetomo, "Distributed denial of services attack protection system with genetic algorithms on Hadoop cluster computing framework", in *Proc. of IEEE Congr. on Evolut. Comput. CEC 2015*, Sendai, Japan, 2015, pp. 1575–1580 (DOI: 10.1109/CEC.2015.7257075).

[11] A. Alsirhani, S. Sampalli, and P. Bodorik, "DDoS attack detection system: utilizing classification algorithms with Apache Spark", in *Proc. 9th IFIP Int. Conf. on New Technol., Mobil. and Secur. NTMS 2018*, Paris, France, 2018, pp. 1–7 (DOI: 10.1109/NTMS.2018.8328686).

[12] O. Osanaiye *et al.*, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing", *EURASIP J. on Wirel. Commun. and Network.*, vol. 2016, article no. 130, 2016 (DOI: 10.1186/s13638-016-0623-3).

[13] A. Bhatia, "Faster detection and prediction of DDoS attacks using MapReduce and time series analysis", in *Proc. Int. Conf. on Inform. Network. ICOIN 2018*, Chiang Mai, Thailand, 2018, pp. 556–561 (DOI: 10.1109/ICOIN.2018.8343180).

[14] S. Hameed and U. Ali, "HADEC: Hadoop-based live DDoS detection framework", *EURASIP J. on Inform. Secur.*, vol. 2018, article no. 11, 2018 (DOI: 10.1186/s13635-018-0081-z).

[15] A. Sharma, C. Agrawal, A. Singh, and K. Kumar, "Real-time DDoS detection based on entropy using Hadoop framework", in *Computing in Engineering and Technology. Proceedings of ICCET 2019*, B. Iyer, P. S. Deshpande, S. C. Sharma, and U. Shiurkar, Eds. *AISC*, vol. 1025, pp. 297–305. Springer, 2019 (DOI: 10.1007/978-981-32-9515-5_28).

[16] C. Wang, T. T. Miu, X. Luo, and J. Wang, "Skyshield: a sketch-based defense system against application layer DDoS attacks", *IEEE Trans. on Inform. Forensics and Secur.*, vol. 13, no. 3, pp. 559–573, 2018 (DOI: 10.1109/TIFS.2017.2758754).

[17] S. Behal, K. Kumar, and M. Sachdeva, "D-FACE: an anomaly based distributed approach for early detection of DDoS attacks and flash events", *J. of Netw. Comp. Appl.*, vol. 111, pp. 49–63, 2018 (DOI: 10.1016/j.jnca.2018.03.024).

[18] K. B. Virupakshar, Narayan D. G., and P. S. Hiremath, "Detection of DDoS attacks in software defined networks", in *Proc. of 3rd Int. Conf. on Computat. Sys. and Inform. Technol. for Sustain. Solut. CSITSS 2018*, Bengaluru, India, 2018, pp. 265–270 (DOI: 10.1109/CSITSS.2018.8768551).

[19] K. B. Virupakshar *et al.*, "Distributed denial of service (DDoS) attacks detection system for OpenStack-based private cloud", *Procedia Comp. Sci.*, vol. 167, pp. 2297–2307, 2020 (DOI: 10.1016/j.procs.2020.03.282).

[20] OpenStack [Online]. Available: https://www.OpenStack.org/

[21] Apache Spark [Online]. Available: https://spark.apache.org

Shweta Gumaste received her B.E. in Computer Science and Engineering from KLE Technological University, Hubballi, India, in 2019. Currently, she is working as associate software engineer in Robert Bosch Engineering and Business Solutions, Bangalore.

E-mail: shwetagumaste@gmail.com
School of Computer Science and Engineering
KLE Technological University
Hubballi, Karnataka, India

Narayan D. G. is currently working as a Professor in the School of Computer Science and Engineering at KLE Technological University, Hubballi, India. He obtained Ph.D. and M.Tech. from Visvesvaraya Technological University, Belgaum in 2017 and 2004 respectively. He obtained B.E. in Computer Science and Engineering from Karnataka University in 2000. He has 20 years of teaching and research experience. His research interest includes wireless mesh networks, cyber security, software defined networks and blockchain. He is a reviewer for many SCIE indexed journals. He has published more than 35 papers in conferences and journals.

https://orcid.org/0000-0002-2843-8931
E-mail: narayan_dg@kletech.ac.in
School of Computer Science and Engineering
KLE Technological University
Hubballi, Karnataka, India

Sumedha Shinde received her B.Sc. and M.Sc. degrees in Mathematics from Karnataka University in 1998 and 2000 respectively. She obtained Ph.D. in Spectral graph theory in 2017 from Visvesvaraya Technological University, Belgaum. Currently, she is working as an Assistant Professor in Department of Mathematics at KLE Technological University, Hubballi, India. Her research interest includes spectral graph theory, application of graph theory in computer networks and applied statistics. She has 20 years of experience and published more than 10 papers in conferences and journals.

https://orcid.org/0000-0002-4154-0025
E-mail: sumedha@kletech.ac.in
Department of Mathematics
KLE Technological University
Hubballi, Karnataka, India

Amit K holds his M.Sc. degree in Computer Science and currently pursuing Ph.D. in the area of network security. He is working as an Assistant professor in KLE Technological University, Hubballi, Karnataka and is having teaching experience of 12 years. His research interest includes network security, artificial intelligence and cloud computing. He has published papers in the area of IoT and teaching pedagogy in international conferences.

E-mail: amitk@kletech.ac.in
Department of MCA
KLE Technological University
Hubballi, Karnataka, India