I. J. Computer Network and Information Security, **2022, 5, 47-59**
Published Online on October 8, 2022 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2022.05.04

Detection of Unknown Insider Attack on Components of Big Data System: A Smart System Application for Big Data Cluster

Swagata Paul
The Assam Kaziranga University/CSE, Jorhat, 785006, India
Techno International New Town/CSE, Kolkata,700156, India
E-mail: swagatapaul@hotmail.com

Sajal Saha
Adamas University/CSE, Barasat, 700126, India
E-mail: sajalkrsaha@gmail.com

Radha Tamal Goswami
Techno International New Town/CSE, Kolkata,700156, India
E-mail: tamal.goswami@gmail.com

Received: 14 September 2021; Revised: 17 December 2021; Accepted: 15 March 2022; Published: 08 October 2022

Abstract: Big data applications running on a big data cluster, creates a set of process on different nodes and exchange data via regular network protocols. The nodes of the cluster may receive some new type of attack or unpredictable internal attack from those applications submitted by client. As the applications are allowed to run on the cluster, it may acquire multiple node resources so that the whole cluster becomes slow or unavailable to other clients. Detection of these new types of attacks is not possible using traditional methods. The cumulative network traffic of the nodes must be analyzed to detect such attacks. This work presents an efficient testbed for internal attack generation, data set creation, and attack detection in the cluster. This work also finds the nodes under attack. A new insider attack named BUSY YARN Attack has been identified and analyzed in this work. The framework can be used to recognize similar insider attacks of type DOS where target node(s) in the cluster is unpredictable.

Index Terms: Unknown Insider Attack, Big Data Cluster, Security Analysis of a Big Data Cluster, Framework for Insider Attack Detection in a Big Data Cluster.

1. Introduction

With the rapid growth of data in the universe, a traditional type of application is no longer suitable for processing large volume and verity of data. Therefore, to process and analyses such huge data, a big data cluster is crucial. A conventional big data cluster comprises of many computers(nodes) interconnected over LAN or WAN. All the nodes in the cluster participate to run applications of big data. As the data volume is huge and size of application is small the application moves to the nodes containing data. The data processing operation must be done in a secure environment. At the same time a big data application should not hamper the execution of other big data application. Therefore, the cluster must be free from any types of attack. Each node of the cluster must be protected from both external and internal attack. The cluster's efficiency can be reduced because of such an attack. To secure cluster nodes engaged in performing a job on big data, we must first assess which processes are being targeted. Different attack detection scheme like DOS/DDoS exists for services like HTTP and other services [1,2,3]. These schemes are not suitable for a big data cluster as the target node(s) of attack is unpredictable. Different IDS exists for anomaly detection which must be extended for big data cluster [4,5]. The limitations of existing works for internal attack detection in a big data cluster are – a) these works are for a fixed target node, b) non availability of attack data set. The objective of this research work focuses on internal attacks generated by the client's big data job. Because of the fact the processes in a cluster are scattered over several nodes, it is important to identify the victim and source of an internal attack in addition to the intrusion.

There are two types of processes that operate on big data clusters - i) processes that are responsible for offering the big data platform, named as big data service (S_{ij} - i^{th} service running on j^{th} node). ii) applications on big data, submitted

by client, named as big data process (P_{ij} - i^{th} process running on j^{th} node). On some nodes, there are identical S_i, while others have unique S_i. It is obvious that all S_{ij} does not participate for a P_{ij}. A group of S_{ij} work together to complete a P_{ij} using the resources of the nodes of the cluster. Some attack process P_{ij} is also executed along with regular P_{ij}. While the process P_{ij} and S_{ij} is in execution on different nodes of the cluster and exchanges data, every network packets are captured from all the nodes and stored them. These packets are further processed to create Data set D1 and D2. these data sets contain attack information. Data set D1 is analyzed to determine which S_{ij} is running on which node N_i. This is the footprint of the cluster. The footprint and the data set D2 are further analyzed to determine the insider attack, source and victim nodes in the cluster. This work detects one new attack named BUSY YARN – which consumes all the cluster resource. Furthermore, nodes affected by this attack has been discovered.

The prime focus of this work is to pinpoint P_{ij} those creating internal attack. Implemented system is compatible with Hadoop based big data platform on Linux OS. In this experimental study Hadoop [6] has been used as a big data cluster with a cluster size of 28 nodes. Hadoop version - 3.1.1 has been installed using Hortonworks Platform [7]. As admin GUI, Apache Ambari [8] is being used. The aim of this research is to present an efficient testbed for the generation of insider attack, data set creation and identify new unknown insider attack. The dupmcap utility of Wireshark Software [9] has been used to capture the network traffics on all the nodes.

In this study, section 2 reports related works. Section 3 depicts the big data cluster generalized architecture. Architecture of the testbed with experiment setup is explained in section 4. Section 5 highlights the methodology. Section 6 is to show the generation of attack. Section 7 describes creation of data sets. Section 8 is for footprint detection from data set D1. Section 9 is for attack detection and analysis using the footprint and data set D2. Section 10, 11, and 12 represents results and discussions followed by conclusions.

2. Related Work

Many academic studies have looked at setting up a testbed and creating a data set for security review of computer network related applications. It's almost impossible to pinpoint the system under attack in a big data cluster with a few to 1000s of computers processing massive amounts of data, particularly if the attack comes from an insider node.

The research [10] describes a fantastic approach for detecting attacks on job schedular component of the cluster running Hadoop version 1.0 which uses Job-tracker and Task-tracker. In newer Hadoop versions, these two elements have been replaced by Resource-Manager and Node-Managers (version 3.1.1). The proposed method analyzes the machine level instructions of the code. For Job-tracker and Task-tracker, the system must be implemented again as this new code contains different pattern of machine level instructions. Furthermore, no data set for attack detection is generated as a result of the work.

According to the research provided in [11,12] , the time it takes for a MapReduce job to complete increases as an attack is created. This is not applicable to other type of job like Spark streaming job. A Spark streaming programme operating on a Hadoop cluster, uses the MapReduce system, and it does not terminate until it is directly terminated. As a result, the programs will flag the streaming job as a kind of attack. New apps are generated by the developers on a regular basis, resulting in the creation of new JAR archives. It is not necessarily the case that they are Java programmes. It may be developed with Python, Spark, or another programming language.

To detect attacks, the work proposed in [13] uses run-time assembly code from java JARs. New big data apps written with java, as well as Scala, Python, and Spark would crash. The system must be extended so that it works for every type of source code.

In [14,15], the authors demonstrate a number of methods for simulating and generating attack data sets for various network-based attacks such as DoS and DDoS. Each of these methods is used for a particular type of attack where target node is known. These are not appropriate for a big data cluster where victim nodes are not fixed.

Project Apache Metron [16,17] project was created with the aim of providing real-time big data protection. It retired at the end of the year 2020.

Hadoop can be set up to run in a protected mode using Kerberos [18], which includes features like authentication, service authorization, etc [19]. However, there are recognized drawbacks to Kerberos [20] and not suitable for internal attacks.

According to the article [21], more than 5K TB data exposed by insecure Hadoop cluster. Cloudera's article [22] details on the new areas. There researchers can collaborate to secure a bigdata platform considering internal attacks.

The analysis discussed in [23] demonstrates how to use cloudlets to build a firewall for a bigdata platform . During service, this firewall operates local to the data source rather than within the cluster.

The attack SEINA presented in [24] is a basic attack which only slow down the performance of a Hadoop cluster. It does not show on which workload the cluster will not be able to schedule a new job and becomes full.

The design and implementation of a new authentication protocol in [25] does detect and prevent all insider attacks which are specially resource consuming.

The vulnerability framework proposed in [26] can't detect the vulnerable big data applications submitted by the users of the big data cluster. It needs for new approaches to detect the flaws by processing these vulnerable big data apps and its logs.

The research on DDoS Attack Detection Using C5.0 in [27] can be tested on a single known host and it is assumed

that the attack payload is same for all attack packets generated by hping3. This work needs a lot of further extension to be applied on a data set generated from a big data cluster where victim(node) of attack is unknown and there may be multiple nodes under attack.

The log bases analysis to investigate application layer DDoS shown in [28] is very efficient and suitable for services like HTTP, FTP, DNS etc. running on a single host as the services generates a lot of meaningful and well formatted log files in the system. A big data cluster logs are scattered in all nodes of the cluster and there are very few similar logs available. Sometimes all logs are not accessible. Thus, the method is not applicable to find these types of attacks in a big data cluster.

Multi-agent based attack detection scheme presented in [29] uses both network traffic and system logs is suitable for a known single host. The work should be extended more for a big data cluster where target node is never known until attack is propagated.

The aim is to build an efficient framework that can create new insider attack, produce data sets for security analysis of a big data cluster, create example data sets for the same and, detect internal attack and the victims of attack in a big data cluster.

3. Architecture for Big Data Cluster

Fig.1 shows a generalized structure for a big data cluster. Besides the Hadoop, new big data systems such as DataTorrent-RTS, Hydra, Google Big-Query, HPCC Systems, Disco, Hydra, Ceph, and others are being developed. For our tests, we used the Hadoop-Ecosystem as a processing platform because of its world known features and advantages. Hadoop big data cluster consists of one or multiple master nodes and multiple slave nodes. A set of master and slave node runs a component like HDFS, YARN etc. Each component is responsible for a service to client application. A component consists of a set of system processes running on different nodes. Therefore, the architecture is made up of two distinct collections of processes - S_{ij} and P_{ij}.

As an example, in Fig.1, The component HDFS manages the data storage in the cluster. HDFS consists of – a) one/two 'NameNode' process running on node MasterB1/B2 and b) multiple 'DataNode' process running on SlaveNodes. A set of unites - S_{ij} (NameNode process and DataNode process) are responsible for data storage and processing. Another set of S_{ij} is responsible for scheduling the client jobs. 'Resource-Manager' process running on node, Master A and 'Node-Managers' processes running on Slave nodes, provides a component called YARN. The component YARN is used to schedule big data client applications. Other S_{ij}'s in the cluster provides the components - MapReduce, ZooKeeper, HBase, Hive, Spark etc. The documentation is given here [30,31].

The second set is P_{ij}, submitted by end users of the cluster. In the Fig.1, "Big Data App 1", "Big Data App 2" etc. are big data client applications. To complete a data processing task, the P_{ij} applications communicate with one or more S_{ij} via LAN/WAN. The communications – a) from client process P_{ij} to system process S_{ij} and b) from system process S_{ij} to system process S_{ij} creates a huge network packet flow among the nodes in the cluster. These network packets contain useful information on various types of communications in the cluster.

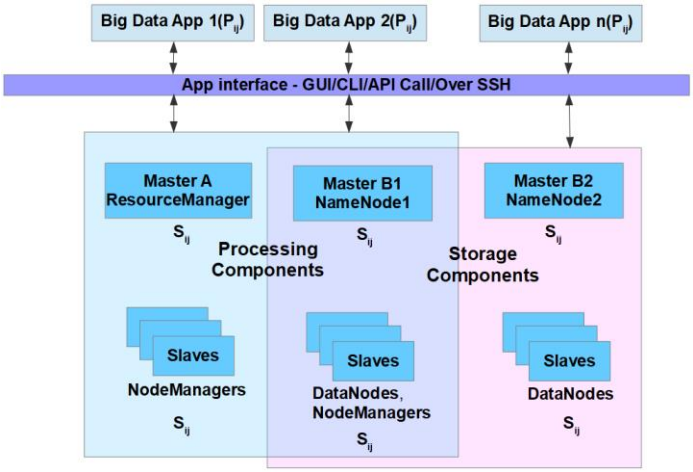

Fig.1. Generalized Big Data Framework

4. Architecture of Testbed and Experimental Setup

Fig.2 depicts the testbed design, attack creation, data set formation, and data set clustering for attack detection. Table 1 shows the cluster specifications. We apply experimental P_{ij} containing attack in addition to standard P_{ij} for our experiment. The attack launcher (AL) node is used to fire these attacks. Network traffic is collected from all the nodes in the cluster when the cluster is in attack with attack process. From the capture center (CC) node, the network traffic

capture process is triggered. The captured traffic is momentarily saved as a .pcapng file on cluster nodes. The CC node then gathers all of this network traffic. The traffic data produced by the experimental processes is then combined and filtered to aid in the identification of attacks. The result is .csv files representing two data sets - data set D1 and data set D2. Data set D1 is being used to detect the cluster footprint. Data set D2 is more specific to TCP conversations. It is used to detect the attack. Starting from traffic capture to attack detection the entire process is in online mode.

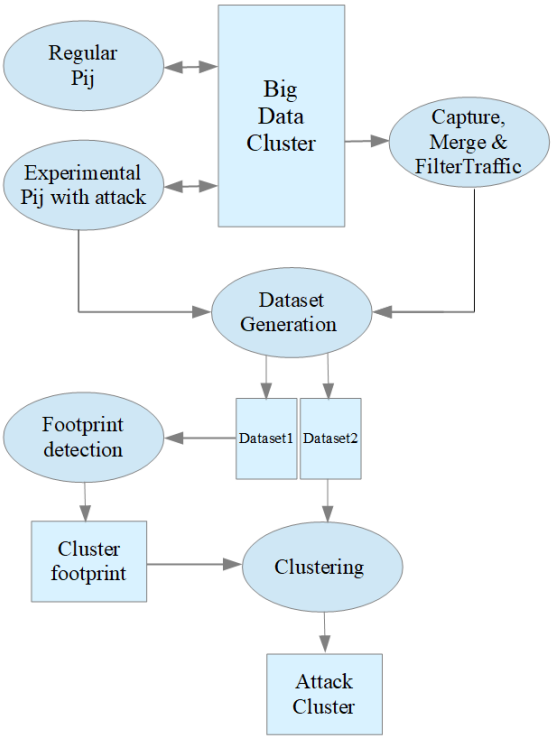

Fig.2. Testbed Architecture

The Hadoop cluster component details are given in Table 1. The operating system and hardware configuration of the cluster nodes are given in Table 2.

Table 1. Details of Cluster Configuration

Sl#	Item	Specifications
1	Ambari	Version: 2.7.3.0, on N1
2	Hadoop File System	Version 3.1.1, 128MB Block, Replication Factor: 3, 13.7 TB HDFS Space
3	NameNode	HDFS Count: 1, on N2
4	DataNodes	HDFS Count: 25, Hostnames: N4 to N28.
5	Hadoop YARN	Version: 3.1.1
6	YARN ResourceManager	Hostname: N2
7	YARN NodeManagers	Count: 24, on N5 to N28
8	Cluster Memory - YARN	Size: 142 GB
9	Minimum Container Size	512MB
10	Maximum Container Size	2GB
11	Min. no of VCores per Container	Count: 1
12	Max. no of VCores per Container	Count: 3
13	% of Vcores Allocation	Percentage of physical CPU allocated for all containers: 80%
14	Apache Spark	Ver 2.3.2
15	Client Library	On all nodes: cseoslab001 to cseoslab028

5. Methodology

Starting with an idle and active cluster as specified in Table 1, the steps followed to conduct this research is given below –

- Attack generation
- Capture network traffic
- Create Data set
- Cluster footprint detection
- Analysis of data set with cluster footprint
- Find attack signature and validate with system matrix

Detailed work has been described in the following sections.

Table 2. Cluster Nodes - OS, Hardware, and Network

Sl#	Item	Specifications
1	OS	Ubuntu 18.04 LST, 64 bits
2	CPU	Intel, Core i5
3	Primary Memory	DDR3, Size: 8GB (in cluster: 6GB)
4	Secondary Memory	SATA, Size - 1TB. A partition of size 700GB is used to store app and data
5	LAN Details	Network Adapter of all nodes - 1Gbps, Network Switch -1Gbps
6	IP Address Range	192.168.10.171-192.168.10.198
7	OS Hostnames	cseoslab001, cseoslab002, ... cseoslab028

6. Attack Generation

YARN is a big data component in Hadoop that includes a ResourceManager(S_{ij}) and multiple ApplicationMaster(S_{ij}). It is used for scheduling the big data jobs(P_{ij}) and to manage the resources of the big data cluster [21]. In this work a new attack has been created to make the component YARN busy in such a way that it can't serve further job requests. The attack creates several new SPARK applications (P_{ij}) using SPARK API and loads HdfsWordCount job in each application. The attack is generated from command line of AL node. The YARN S_{ij} works together to schedule all such P_{ij} and acquire all the resources of the cluster requires for YARN. A Spark job is not terminated by itself until it is killed explicitly. Now the cluster does not have any resource to schedule new job, but it accepts new jobs and can't schedule them. The status of such jobs remains unfinished. We call this new attack - BUSY YARN. We can't use iptable or related tools to block packets from AL node assuming it as a DOS type of attack. Because there will be huge number of valid communications among different S_{ij} used for the component YARN running on several N_i. We need to find out the communication pattern for these new unknown insider attack. The attack - BUSY YARN has been executed several times as a batch of 50 P_{ij}, 100 P_{ij} and 200 P_{ij}. Each batch has been executed as a shell script. As Spark jobs are not terminated automatically, after finishing the data capture process the jobs have been killed forcefully to make the cluster ready for next jobs.

Attack generation code using Linux shell script is shown in Table 3. We have used a) YARN REST API with curl command to create an YARN app, b) Then prepared a json file containing spark streaming job details with HdfsWordCount module deployed in cluster mode c) Then again call YARN REST API with curl to apply the json. The attack generation script is shown in Table 3.

7. Data set Creation

A data set for insider attack detection in a big data cluster is the commutative network traffic captured on individual host. In Fig.3 attack launcher node, capture center node and different big data cluster nodes are shown. In respective nodes we run various processes. These processes help to capture and collect network traffic, generate attack etc.

- On CC Node (CC): -
  - ✓ Capture-manager process, denoted as 'M'.
- On Attack-launcher Node(AL):-
  - ✓ Attack-launcher process, denoted as 'L'.
- On Cluser Nodes (MN, SMN, SN):-
  - ✓ Capture-agent processes denoted as 'A'.

Table 3. Attack Generation Code

Sl#	Item	Specifications
1	Create YARN app	t='text' appid=`curl -v -X POST http://cseoslab002:8088/ws/v1/cluster/apps/new-application?user.name=hdfs 2>/dev/null | sed -e 's/[{}]/"/g' | awk -v k="$t" '{y=split($0,x,","); for(k=1; k<=y; k++) print x[k]}' | head -1 | cut -d ":" -f 2`

2	Generate json file	json1='\{
		"application-id":myappid,
		"application-name":"BusyYarn-SPARK-api_by_spaul",
		"application-type":"SPARK",
		"am-container-spec": {
		"commands":{
		"command":"spark-submit --name BUSY-YARN-Attack2021-Spaul --
		class org.apache.spark.examples.streaming.HdfsWordCount --deploy-mode cluster
		/usr/hdp/current/spark2-client/examples/jars/spark-examples_2.11-2.3.2.3.1.0.0-78.jar
		/spaul/in01"
		}
		}
		}'
		
		srt1="myappid"
		json2=`echo $json1 \| sed -e "s/myappid/$appid/g"`
		echo $json2 > newfile.json

3	Apply the json to new job	opt="-v -X POST" opt2="application/json" curl $opt --data-binary @newfile.json -H "Content-type: $opt2" http://cseoslab002:8088/ws/v1/cluster/apps?user. name=hdfs

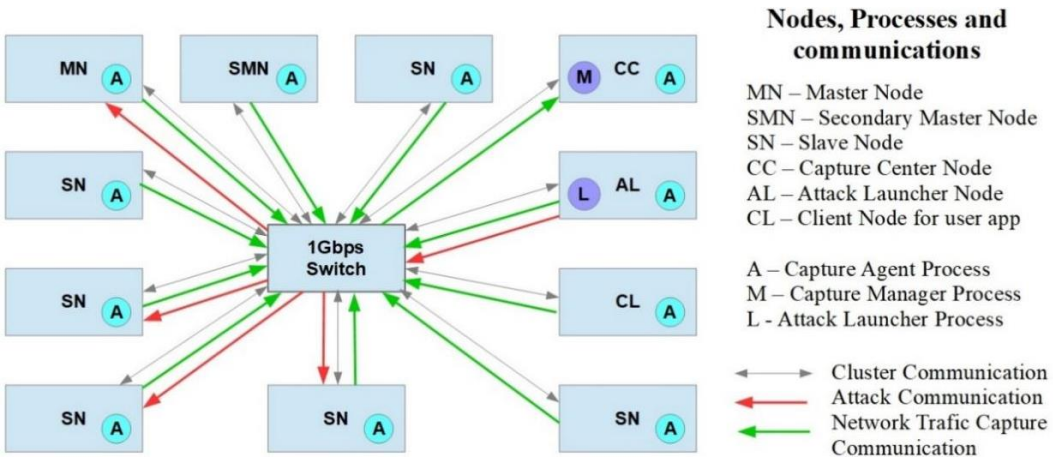

Fig.3. Processes and Communication on different nodes in the cluster

The flow of operations for data set creation using network traffic captured on different node is presented in Fig.4. The operation sequences are –

- The manager 'M' sends begin-capture signal to all the capture agents 'A' running on the nodes of the cluster;
- network traffic capture operation starts in all nodes by process 'A';
- Experimental P_{ij} process started by CC node;
- BUSY YARN attack is send to the cluster by process 'L' from AL node;
- Attack job and other big data processes executed for some time;
- The manager 'M' sends end-capture() signal to all the agents 'A';
- Process 'A' finish the capture operation;
- Process 'A' stores the traffic in local file system on each node;
- Process 'M' send get-data signal to all 'A' in the cluster;
- All 'A' processes send the capture network traffic data to 'M';
- 'M' receives the data on CC node;
- After receiving 'M' performs the data filtering operation, merge the traffic data, remove duplicate packets;
- CC node also receives the attack data;
- CC node creates two data set as D1 and D2 in .csv format.

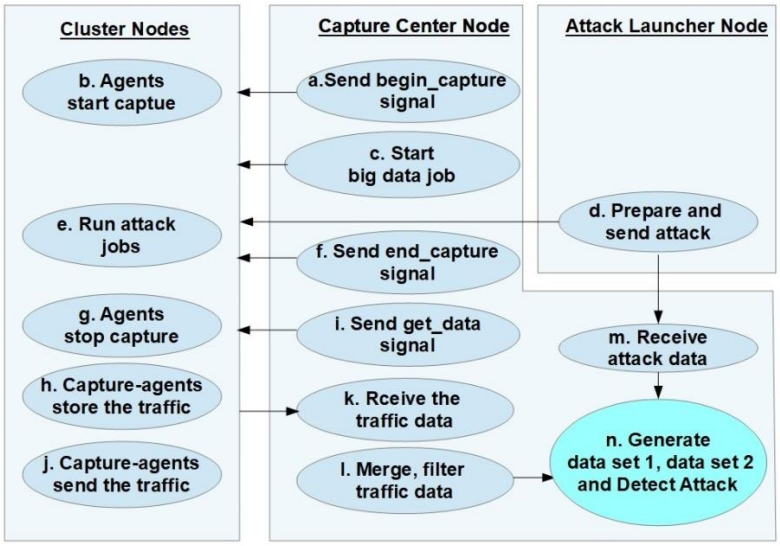

Fig.4. Attack Generation and Data Capture Flow

The data set D1 is little bulky in size. Therefore, we do not use it for attack detection directly. Data set D1 is used to detect the footprint of the cluster. This footprint works as first input to the attack detection process. The format of data set D1 is shown in Fig.5.

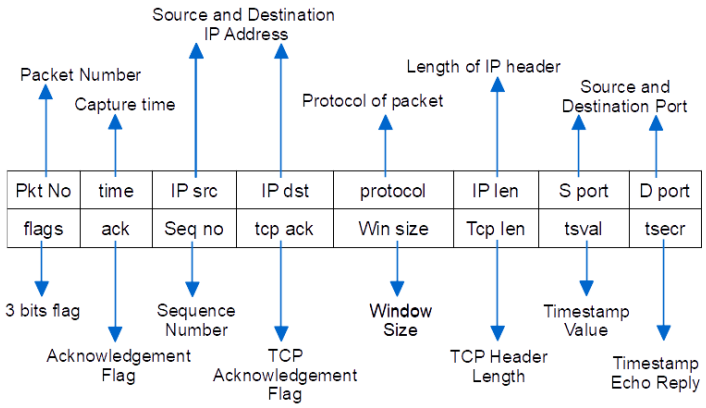

Fig.5. Data set D1 - Record Format

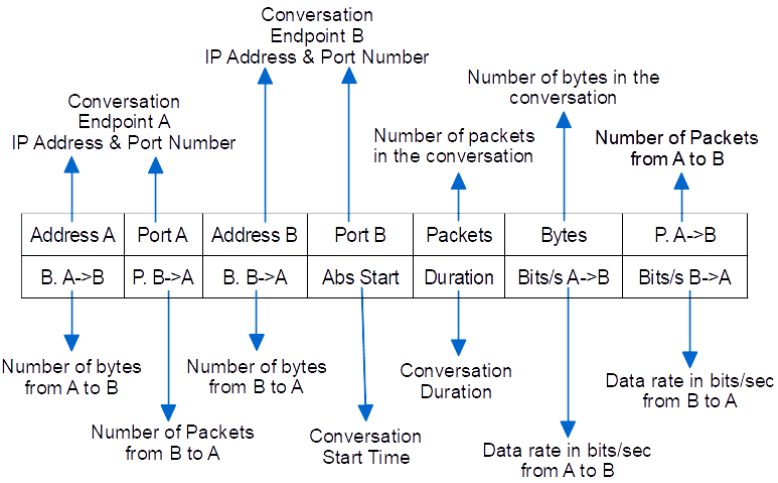

Fig.6. Data set D2 - Record Format

The second input to the attack detection process is data set D2. Data set D2 is a new data set which represents all TCP conversations between two processes (S_{ij} and/or P_{ij}) running on two N_i among the big data cluster nodes. TCP conversations are distinguished by searching the three-way handshakes in records of data set D1, i.e., by looking the SYN and SYN-ACK packets from each record.

To find SYN packets the following filters are applied (Filter1): with tcp.flags - .syn = 1 && .ack = 0.

To find SYN-ACK packets, the following filters are applied (Filter2): with tcp.flags - .syn = 1 && .ack = 1.

These two filters are combined with logical OR: Filter1 | | Filter2.

The format data set D2 is shown in Fig.6. Here A and B are two cluster nodes participated in a TCP conversation.

For the purpose of insider attack - BUSY YARN detection, we created three sets of data set D1 and data set D2. For each data set the attack been executed in a batch as a batch of 50 P_{ij}, 100 P_{ij} and 200 P_{ij}. The detail of each data set is shown in Table 4.

Table 4. Data set D1 and Data set D2 details

Sl#	# of attacks in a batch	Duration of Capture	# of records in D1	# of records in D2
1	50	2:58min	419150	2405
2	100	2:52min	486835	2482
3	200	2:15min	390033	2709

8. Footprint Detection

Detection of each S_{ij} was done by analyzing the data set D1 on CC node. The editcap utility in Wireshark was used to eliminate duplicate network packets and unnecessary records of the types ICMP, DNS query, Broadcast packets etc., and then the file was translated to .csv. The records in D1 are sorted by packet timestamp. The S_{ij} like NodeManagers, ResourceManager, Scheduler, Timeline Service, History Server etc. are the components of YARN. These are in charge of creating the P_{ij} and resources allocation to P_{ij}. The details work has been shown in [32]. The outcome which is necessary for Busy YARN attack detection is the information about Resource Manager WebUI.

As per the footprint detection process, this manager process is running on node N_2 using port number 8088.

9. Attack Detection and Analysis

We have used K means Clustering algorithm to create clusters from data set D2 which has been generated by running 3 batch of 50, 100, 200 Spark job. The reason for choosing K means for the purpose of Busy YARN attack detection experiment are –

- it is unsupervised learning,
- It is computationally faster than hierarchical clustering for small k,
- it can be used to find groups which have not been explicitly labeled in the data, and
- it can scale to large data sets.

Count of 'Address B' and count of 'bytes' are the input to the algorithm. Selection of number of clusters is automatic. The clusters and underlying details are shown in Fig.7, Fig.8, and Fig.9, which are similar. The clusters formed by K means are shown as circles in left side of each figure. The circles which are big in size has more similar repetitive conversation than the smaller ones. In right side, after applying the footprint (as found in footprint detection), the details of largest(C1) and 2nd largest(C2) clusters shown. Here,

Number of packets in a conversation is labeled as ‘packet1’ which is 10 and 12 in C2 and C1 respectively.

“Bytes A to B” and “Bytes B to A” are amount of data bytes in each of the 2 cluster: C2(981, 787) and C1(600, 1204)

Count of address B from address A, i.e number of similar conversations are 50, 100 & 200 respectively in Fig.7, Fig.8, and Fig.9. These denotes the repetitive jobs.

The details of cluster analysis are shown in Table 5.

Table 5. Analysis of Data set Clusters

Sl#	# of attacks in a batch	# of Conversation in Data set D2	# of Cluster	Main Large Clusters	# of Packets in conv. 1st Cluster(C2): A to B, B to A	# of Packets in conv. 2nd Cluster(C1): A to B, B to A	# of Bytes 1st Cluster(C2) A to B, B to A	# of Bytes 2st Cluster(C1) A to B B to A
1	50	2405	6	3, 5	10: 6, 4	12: 7, 5	981, 787	600, 1204
2	100	2482	5	3, 4	10: 6, 4	12: 7, 5	981, 787	600, 1204
3	200	2709	4	3, 4	10: 6, 4	12: 7, 5	981, 787	600, 1204

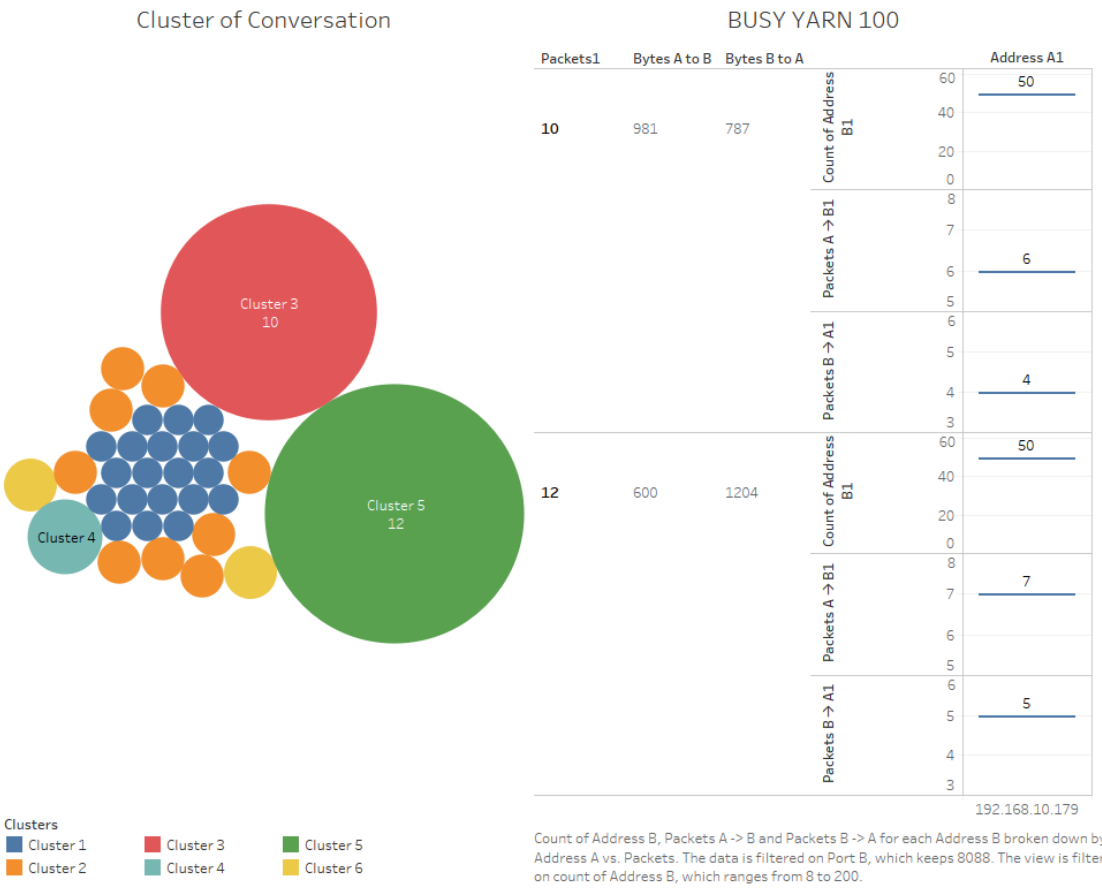

Fig.7. BUSY YARN - 50

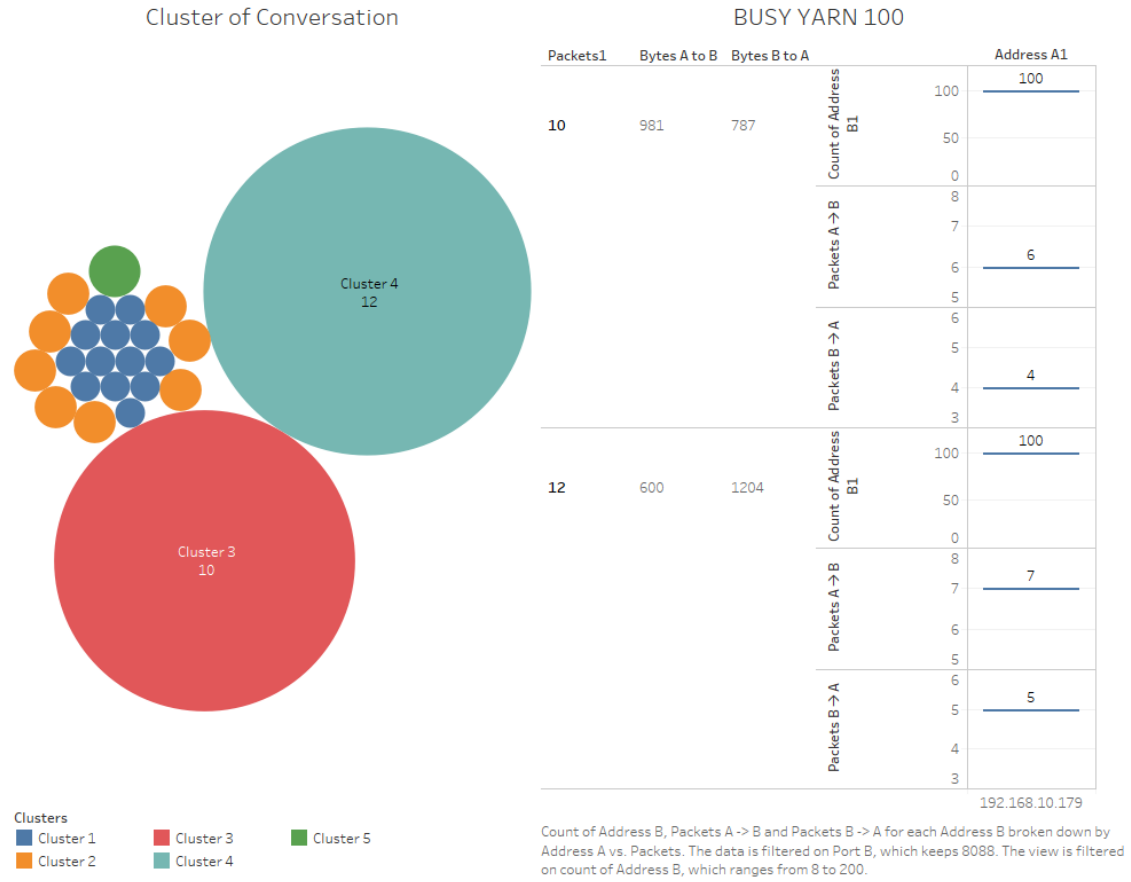

Fig.8. BUSY YARN – 100

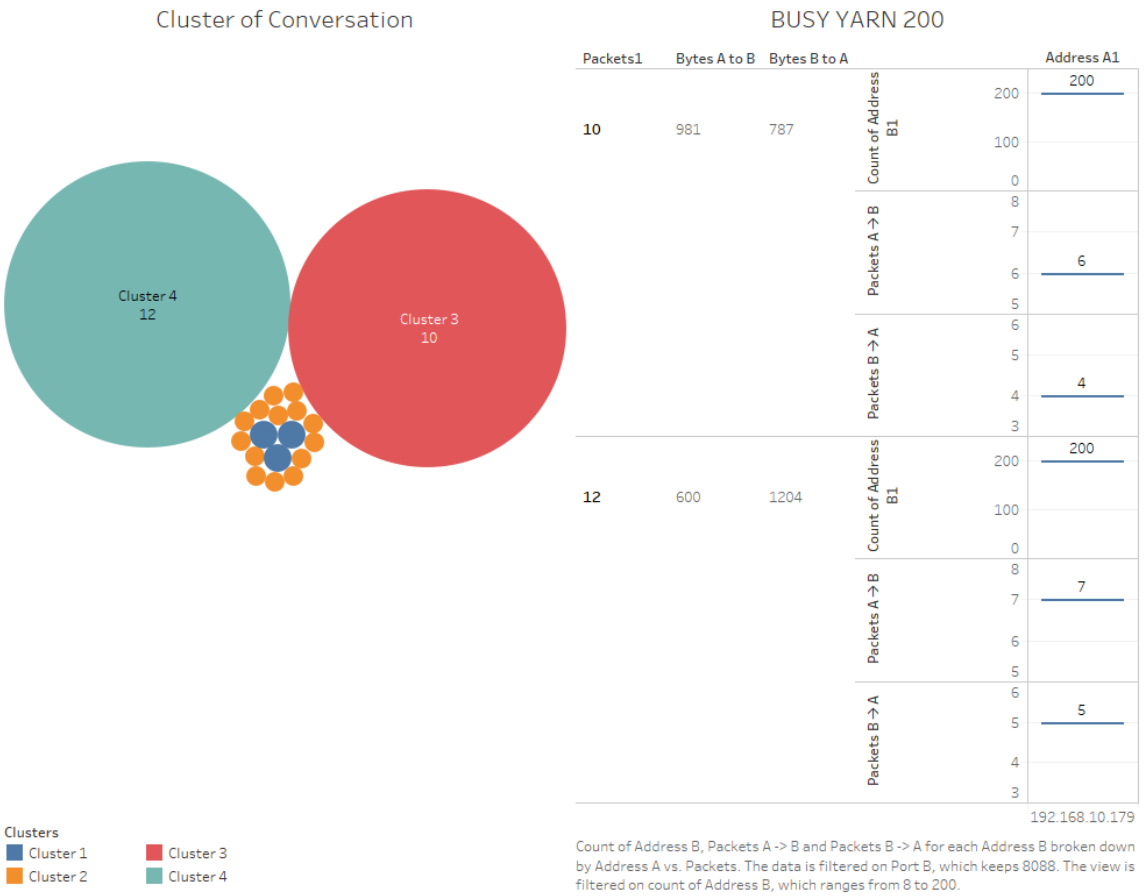

Fig.9. BUSY YARN - 200

After analyzing the data set D2 with the help of footprint (obtained using data set D1), the corresponding resource usage of the cluster is obtained from the big data cluster. This resource usage has been recorded after 2hrs of job submission. This is shown in Table 6. This resource usage statics remains same for long time. Number of completed job always remains zero.

Table 6. Busy YARN Attack Apps vs YARN Resources

Sl#	# of attacks in a batch	Apps Submitted/ Pending	Apps Running/ Completed	Containers Running	Memory Used/Total(GB)	VCores Used/Usable
1	50	100/50	50/0	50	28/144	50/57
2	100	189/170	19/0	57	85.5/144	57/57
3	200	254/199	55/0	57	32/144	57/57

Other regular non attack P_{ij} also has been submitted. Their completion status improves over time and after certain time, depending on the application type all are completed. For the specific attack BUSY YARN, K means always created two big clusters for all 3 sets of data. These two are the main clusters.

As per Table 5, the observations are –

- the number of packets in a conversation is 10(where 6 numbers of packets are from A to B and 4 numbers of packets are from B to A) and 12(where 7 numbers of packets are from A to B and 5 numbers of packets are from B to A) for 1st(C2) and 2nd(C1) cluster respectively.
- The number of bytes in the conversations in the main clusters (C1 and C2) are fixed for all 3 sets of data - C2(981, 787) and C1(600, 1204).

Therefore, these numbers form a pattern for this specific attack BUSY YARN.

As per cluster configuration in Table 2 there are 24 NodeManagers (S_{i5} to S_{i28}) in the cluster. Each NodeManagers can use max. 3 containers, 1 container per vcore. Therefore, maximum number of possible containers are 24x3=72, furthermore, 80% these containers are usable. Thus, number of usable containers = 72 x 80% = 57. That is 57 containers are usable for client apps as per settings in YARN.

Sl no #2 and #3 in Table 6 shows that all the usable containers are running. So, the system can't admit more apps

as the existing containers are never released. Therefore, the attack Busy YARN creates a deadlock in the cluster. It is also observed that, although reasonable amount of cluster memory is free, as containers are not free, new applications can't be created. Thus, the attack is validated.

10. Discussion on Analysis

The result obtained from data set D2 with the footprint of the cluster received from data set D1, as given in Table 5 and Table 6 clearly shows that for the internal attack Busy YARN –

- There are specific patterns in the network traffic conversation.
- During and after attack, all the usable containers has been allocated for the attack jobs, thus the YARN component becomes fully BUSY.

K means algorithm uses Count of 'Address B' and count of 'bytes' from records of data set D2. These conversation clusters are filtered using the result of footprint. Thus, it does not show all other conversation clusters, which may have huge number packets, but those are not part of YARN component.

The existing cluster monitor can't detect that these attack jobs, running for long time, will not complete until they are terminated forcefully. The big data cluster may be configured in a way such that the long running jobs will be terminated automatically after a certain timeout. In that case some other vital jobs will also be terminated. So, the attack jobs which are running for long time can't terminated in this way. This work detects the source of the attack and also detect that the cluster does not take further load from new apps and the current attack apps will not finish in future.

This system application for a big data cluster can be used by the cluster admin. The cluster admin can run this to detect the performance reduction as well as internal attack detection. The cluster may be a native Hadoop cluster or it may be other big data cluster based on Hadoop, like Cloudera.

11. Conclusions

The mechanisms or tools available for testbed design for internal new attack generation, data set generation using network traffic, and new attack detection method were investigated in this article. Since separate node is used to submit attacks (AL node) and process network traffic (CC node), these tools have no effect on Hadoop Cluster performance. We used the K means algorithm for data set analysis in order to detect attacks because of its well-known benefits. It does not use any additional resources from the Hadoop cluster node's operating system. We either collect continuous network traffic or capture it frequently, depending on the necessity. The visualizations of the pattern of attack programme actions were created using Tableau version 2020.4. In the work, we created a system programme. This can be installed in a Hadoop based cluster with minimum effort. Once installed and configured, this can be used to generate data set from big data cluster and find the BUSY YARN Attack pattern and the nodes under attack. As we identify the nodes under attack, it is easier by a system admin to a) kill the tasks which are making the system busy or b) isolate the node from the rest of the cluster. Thus, this system tools provides a great help for the big data cluster system admin. For further research, this tool can be extended by the researchers to identify other internal attack patterns, the nodes under attack and the nodes generating the attacks. The framework can be used to detect other internal attack like detection of massive non useful file I/O.

12. Acknowledgment

This study was carried out at Big-Data Laboratory at Techno International Newtown (TINT), Kolkata, India. Tableau Academic Programs provided us with a technical version of Tableau Software.

References

[1] M. Aamir and S. M. A. Zaidi, "DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation," *Int. J. Inf. Secur.*, vol. 18, no. 6, pp. 761–785, Dec. 2019, doi: 10.1007/s10207-019-00434-1.

[2] J. Jiao et al., "Detecting TCP-Based DDoS Attacks in Baidu Cloud Computing Data Centers," in 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, Hong Kong, Sep. 2017, pp. 256–258. doi: 10.1109/SRDS.2017.37.

[3] G. Shrivastava, P. Kumar, B. B. Gupta, S. Bala, and N. Dey, Eds., Handbook of Research on Network Forensics and Analysis Techniques: IGI Global, 2018. doi: 10.4018/978-1-5225-4100-4.

[4] T. H. Divyasree and K. K. Sherly, "A Network Intrusion Detection System Based On Ensemble CVM Using Efficient Feature Selection Approach," *Procedia Comput. Sci.*, vol. 143, pp. 442–449, 2018, doi: 10.1016/j.procs.2018.10.416.

[5] J. Jabez and B. Muthukumar, "Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach," *Procedia Comput. Sci.*, vol. 48, pp. 338–346, 2015, doi: 10.1016/j.procs.2015.04.191.

[6] D. Cutting, "The Apache™ Hadoop®." May 2019. [Online]. Available: https://hadoop.apache.org

[7] Hortonworks, "Apache Hadoop Ecosystem and Open Source Big Data Projects." May 2019. [Online]. Available: https://hortonworks.com/ecosystems

[8] Ambari, “The Apache Ambari Project.” May 2019. [Online]. Available: https://ambari.apache.org
[9] Wireshark, “dumpcap - Dump network traffic.” Jul. 2019. [Online]. Available: https://www.wireshark.org/docs/man-pages/dumpcap.html
[10] S. K. Aditham and N. Ranganathan, “SYSTEMS AND METHODS FOR DETECTING ATTACKS IN BIG DATA SYSTEMS,” 20190089720, Mar. 2019 [Online]. Available: http://www.freepatentsonline.com/y2019/0089720.html
[11] W. Glenn and W. Yu, “Cyber Attacks on MapReduce Computation Time in a Hadoop Cluster,” in *Big Data Concepts, Theories, and Apps*, Springer, 2016, pp. 257–279.
[12] J. Huang, D. M. Nicol, and R. H. Campbell, “Denial-of-service threat to Hadoop/YARN clusters with multi-tenancy,” in *2014 IEEE International Congress on Big Data*, 2014, pp. 48–55.
[13] S. Aditham and N. Ranganathan, “A system architecture for the detection of insider attacks in big data systems,” *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 974–987, 2017.
[14] S. Alzahrani and L. Hong, “Generation of DDoS Attack Dataset for Effective IDS Development and Evaluation,” *J. Inf. Secur.*, vol. 9, no. 04, p. 225, 2018.
[15] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, “Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling,” *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, 2017.
[16] Metron, “Apache Metron.” May 2019. [Online]. Available: https://hortonworks.com/apache/metron
[17] Metron, “Apache Metron Big Data Security.” [Online]. Available: https://metron.apache.org
[18] MIT, “Kerberos - Network Auth. Protocol.” [Online]. Available: https://web.mit.edu/kerberos/
[19] Hadoop, “Hadoop in Secure Mode.” Jul. 2019. [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SecureMode.html
[20] MIT, “Kerberos Drawbacks.” [Online]. Available: https://web.mit.edu/rhel-doc/5/RHEL-5-manual/Deployment_Guide-en-US/ch-kerberos.html
[21] S. Khandelwal, “Insecure Hadoop Clusters Expose Over 5,000 Terabytes of Data.” [Online]. Available: https://thehackernews.com/2017/06/secure-hadoop-cluster.html
[22] M. Yoder and S. Acharya, “Protecting Hadoop Clusters From Malware Attacks.” Nov. 2018. [Online]. Available: https://blog.cloudera.com/blog/2018/11/protecting-hadoop-clusters-from-malware-attacks
[23] C. R. Panigrahi, M. Tiwary, B. Pati, and H. Das, “Big data and cyber foraging: future scope and challenges,” in *Techniques and Environments for Big Data Analysis*, Springer, 2016, pp. 75–100.
[24] J. Wang, T. Wang, Z. Yang, Y. Mao, N. Mi, and B. Sheng, “SEINA: A stealthy and effective internal attack in Hadoop systems,” in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 525–530. doi: 10.1109/ICCNC.2017.7876183.
[25] Z. Dou, I. Khalil, A. Khreishah, and A. Al-Fuqaha, “Robust Insider Attacks Countermeasure for Hadoop: Design and Implementation,” *IEEE Syst. J.*, vol. 12, no. 2, pp. 1874–1885, 2018, doi: 10.1109/JSYST.2017.2669908.
[26] Hakan Kekül, Burhan Ergen, Halil Arslan, "A New Vulnerability Reporting Framework for Software Vulnerability Databases", International Journal of Education and Management Engineering (IJEME), Vol.11, No.3, pp. 11-19, 2021. DOI:10.5815/ijeme.2021.03.02
[27] Hariharan. M, Abhishek H. K, B. G. Prasad, "DDoS Attack Detection Using C5.0 Machine Learning Algorithm", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.9, No.1, pp. 52-59, 2019.DOI:10.5815/ijwmt.2019.01.06
[28] T. Raja Sree, S. Mary Saira Bhanu, "Investigation of Application Layer DDoS Attacks Using Clustering Techniques", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.8, No.3, pp. 1-13, 2018. DOI:10.5815/ijwmt.2018.03.01
[29] Xin ZHANG, Ying ZHANG, Raees ALTAF, Xin FENG, "A Multi-agent System-based Method of Detecting DDoS Attacks", International Journal of Computer Network and Information Security(IJCNIS), Vol.10, No.2, pp.53-64, 2018. DOI:10.5815/ijcnis.2018.02.07
[30] D. Cutting, “Apache Hadoop 3.1.1 Documentation.” Nov. 2021. [Online]. Available: https://hadoop.apache.org/docs/r3.1.1/
[31] A. S. Foundation, “Apache Hadoop 3.3.0 - Apache Hadoop YARN.” Jun. 2020. [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
[32] S. Paul, S. Saha, and R. T. Goswami, “Big Data Cluster Service Discovery: A System Application for Big Data Cluster Security Analysis,” in *Data Science and Analytics*, Springer Singapore, 2020, pp. 331–341. doi: 10.1007/978-981-15-5830-6_28.

Authors’ Profiles

Mr. Swagata Paul did his B.E. from Asansol Engineering College, Asansol, West Bengal, India in Information Technology in the year 2002. He completed his MTech from Maulana Abul Kalam Azad University of Technology, West Bengal (formerly WBUT), India in the year 2005. He is pursuing his PhD from The Assam Kaziranga University, Assam, India. He has 18 years of experience in teaching. He is working at as Assistant Professor, Department of CSE, Techno International New Town, Kolkata, India. Total number of research publications are 7.

Dr. Sajal Saha has 18 years of academic experience working both in the affiliated college and university system. Currently he is the Professor and Head, Department of Computer Science and Engineering and Director-Product & Innovation of Adamas University. Prior to this, he was the Principal of Meghnad Saha Institute of Technology under Techno India Group affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal. From 2019 to 2021, he served as Dean of the School of Computing Sciences and Dean of Research at The Assam Kaziranga University. There he served as Associate Dean (Compliance) from 2016 to 2019. His total number of research publications are 40. He authored 2 books titled "GIS and Remote Sensing: Applications in Flood Damage assessment" and "Mobility Management in IP based Network: Framework, Issues and Challenges". For more details: https://sites.google.com/site/sajalsahaofficial/home

Dr. Radha Tamal Goswami is the Director of Techno International New Town, Kolkata and Professor in the Department of Computer Science and Engineering. He is having 25 years of experience in the field of academics and research and administration. He was the professor in Computer Science and Engineering and also worked as Director BIT Mesra Kolkata Campus since 1995. His research interest is in the field of Network Security and BigData Security and Analytics. He is the visiting faculty of 10 Institutions and member of ACM, IEEE, CSI and NIPM. He has published almost 50 research papers. He chaired many National and International conferences. He is the academic and BOG member of Kaziranga University, Andhra University, Anna University, NIPM.

How to cite this paper: Swagata Paul, Sajal Saha, Radha Tamal Goswami, "Detection of Unknown Insider Attack on Components of Big Data System: A Smart System Application for Big Data Cluster", International Journal of Computer Network and Information Security(IJCNIS), Vol.14, No.5, pp.47-59, 2022. DOI:10.5815/ijcnis.2022.05.04