Leveraging Big Data Analytics for Real-time DDoS Attacks Detection in SDN

G. Dileep Kumar, Research Scholar, Jawaharlal Nehru Technological University, Hyderabad, India, dileep.gdk@gmail.com

Dr. CV Guru Rao, Professor, Department of CSE, SR Engineering College, India, guru_cv_rao@hotmail.com

Abstract **A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the targeted server, service or network with a flood of huge network traffic. DDoS attacks have been increasing in volume, complexity and damage. This pace of network traffic generation has created new set of challenges and motivated to find new methods to defend. Traditional data analysis methods have difficulties in defeating these attack by taking substantial amount of time. The security of Software Defined Network (SDN) is a big concern. SDN can expose the network to vulnerabilities on its three layers and the impact of attacks are much severe when compared to conventional networks. Big Data Analytics plays a major role as the network logs are extremely large which could not be handled on a single server machine that has limited computing resources. Leveraging Big Data Analytics enables to analyze large volumes of disparate and complex data from different sources in different formats.**

Hence, the study proposes an architecture of combining Big Data and SDN. The study adopts Support Vector Machine (SVM) and proposes two novel methods for attack detection. The results show good detection ratio of proposed methods over SVM.

Keywords —DDoS Attack, Big Data, IDS, SDN, Detection, Analytics

I. INTRODUCTION

A Distributed Denial of Service (DDoS) attack is malicious attempt to make victim machine or web server and network resource unavailable to intended users by temporarily disrupting the services of a host machine or network resource [1]. DDoS attacks have been increasing in volume, complexity and damage. Banks and other financial institutions experience a high share of DDoS attacks in recent years [4]. DDoS attack is the complicated and more powerful version of Denial of Service (DoS) attack in which many compromised machines are involved. Attacker uses botnet to launch attack. A botnet is a network of compromised machines or bots, also called slaves or zombies all around the world. Botnet has a command & control infrastructure that could be used for various malicious activities. Botnets have a wide range of purposes including DDoS attacks, email spam delivery, password cracking, key logging, crypto currency mining etc. Bots could automatically scan entire network and propagate themselves using known vulnerabilities and weak passwords on other machines. Once a machine is compromised, a small program is installed for future activation by the attacker. An attacker can instruct the bots in the network to execute actions at a certain time such as sending large number of requests to a target server so that it would be unable to serve requests by legitimate users resulting in DDoS attack.

Generally, DDoS Attack consists of the following 4 elements as shown in figure 1:

1. Attacker, is the person who launches the DDoS attacks.
2. Controllers, are compromised machines on which the attacker running malicious code to control large set of zombies.
3. The Zombies or slaves are also compromised systems which are running malware to generate large set of attack packets and send them to intended victim.
4. Victim machine.

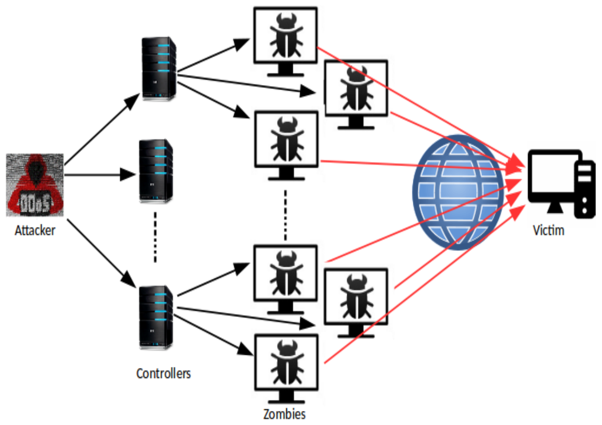


Figure 1: General DDoS Architecture

Software Defined Networking (SDN) brings programmability, automation and superior control over the network in order to increase the scalability and flexibility. In SDN, the processing of the network packets is not done by the switch. Thus SDN architecture decouples the network control from the data or forwarding plane which consists of network devices forwarding traffic based on the control plane or SDN controller policy. This simplifies the design of new protocols and implementation of new network services. The SDN controller makes the decision of whether the packet would be forwarded by the switch or would be dropped and appropriately it makes a flow entry in the flow table of the switch. OpenFlow is one of the protocols that could be used for communication between the data plane and the control plane [2] [3].

The security of SDN is a big concern. SDN can expose the network to vulnerabilities on three layers of it and the impact of attacks are more severe when compared to traditional networks. For instance, the logically centralized SDN controller can lead to many security challenges as if it becomes unavailable then the whole SDN architecture is lost. DDoS attack are the one which makes the controller unavailable to its own network. Whenever DDoS attack is launched, every packet will be sent to the controller for decision making or processing which would exhaust its computing resources and thus the controller could become unavailable for processing of legitimate packets. Moreover multiple flow entries will be made in the flow table of switch which would also be a processing overhead for it [5].

Attacks are increasing in volume, complexity and damage. As per main findings of Cisco's global IP traffic forecast, the annual global IP traffic would pass zettabyte (ZB) by the end of 2017 and it could reach 2.3 ZB per year by 2020 [6]. In September-2016, an IoT botnet built from the Mirai malware, perhaps the largest botnet on record, was responsible for a 600 Gbps attack targeting Brian Krebs's security blog (krebsonsecurity.com) [7]. Mirai's strategy is quite simple; it uses a list of 62 common default usernames and passwords to gain access primarily to home routers, network-enabled cameras, and digital video recorders, which usually have less robust protection than other consumer IoT devices. The same month, a Mirai based attack against the French webhost OVH broke the record for the largest recorded DDoS attack at least 1.1 Tbps, and perhaps as large as 1.5 Tbps [8]. This pace of data generation motivates researchers to innovate novel methods to mitigate them with high speed. The attacker controls a large set of computers to make requests. The number and the size of network logs are also extremely large, this would run into tens of terabytes i.e. Big Data. So, if the traditional network log analysis method is used for these large network logs, it would take a substantial amount of time to analyze and detect DDoS attacks. As a result, using Big Data technology is really important for this type of task.

The Big Data Analytics [10] is a process of ingesting, storing and analyzing large volume of data to discover hidden patterns and other useful insight. Big data is not defined by how you measure data [9]. The benefits of Big Data include depth, time, storage, processing, analysis, flexibility and usability. The Big Data ecosystem provides set of tools for different tasks such as data ingestion, data integration, data analysis and data visualization. Data ingestion can be done either in real time or batches. Data can be pulled from relational data bases or streamed from web logs. Flume [11], Sqoop and Kafka are well known tools which could be used for data ingestion. The Apache Hadoop [12] is an open-source software framework designed for storage and processing of Big Data over a cluster of commodity hardware based computers. There are many other tools and at times one has to customize as per requirements. NoSQL based tools such as HBase [15], MangoDB [16], Cassandra [17], Spark SQL, Spark streaming [18], Flink and Storm [19] can be used for real time data processing. Once data is processed, one can visualize data using standard reporting tools such as Datameer, d3js, Tableau, Qlikview and many more. Leveraging Big Data analytics and suitable tools enables to analyze large volumes of disparate and complex data from different sources in different formats.

Our paper presents the following contribution: 1) A Big Data and SDN based prototype. 2) Two novels methods for real-time DDoS detection which are presented in section 3. 3). Comparative analysis of proposed methods with existing method, support vector machine (SVM).

The paper is structured as follows: Section II reviews the related literature. Section III explains design and implementation of our proposed approach to detect DDoS attacks. Section IV discusses the results produced as part this work. Finally, Section V concludes our work with an insight to future plan.

II. Related Work

The analysis of malicious network packets and processing of logs for threat detection has been a tedious task for years as attackers are changing their methods and tactics in launching DDoS attacks. Recently, Apache Hadoop and its ecosystem has attracted network security community because of the scalability, simplicity and fault tolerance features [12]. The rest of the section presents where Big Data tools and technology used for attack detection.

Jerome et al. [21] proposed a scalable peer-to-peer MapReduce based detection mechanism, Botcloud which is combination of network and host approaches. The approach first generates large volume of Netflow data and then applies PageRank algorithm to differentiate the dependency connected hosts to detect botnets. The MapReduce based PageRank algorithm showed efficiency on cluster.

Lee et al. proposed Hadoop based detection mechanism which implements HTTP GET flooding detection algorithms using MapReduce on distributed computing platform [22]. The counter based method counts total traffic received or web page request from clients. Based on the threshold value specified, the server is alarmed. The access-pattern based method makes an assumption that the clients infected by the same bot conduct similar behavior so that the attacker can be differentiated from normal client. But the proposed framework supports offline batch processing of traffic traces only.

Apache Hadoop divides data into multiple blocks of same size and distribute them in a cluster that are to be processed independently. But this could dislocate traffic structure. To overcome this problem, Johan et al. [23] proposed Hashdoop, a MapReduce framework for network anomaly detection. The developed hash function divides network traffic into blocks which preserves the special and temporal traffic structures. This leverages MapReduce framework for efficient anomaly detection.

Sufian and Usman [24] proposed HADEC, a Hadoop based Live DDoS Detection framework to handle flooding attacks efficiently using MapReduce. They have implemented counter based DDoS detection algorithm to detect TCP-SYN, HTTP GET, UDP and ICMP attacks only.

SDN provides the ability to have a centralized control and real-time response over a set of routers, and there are a vendors working on integrating DDoS mitigating features into the SDN controllers, such as Brocade and Radware. These applications provide attack detection and control over a whole SDN, but the detection and access rules are still largely packet filtering based.

Mousavi et al. propose a DDoS attack detection scheme in SDNs using the notion of entropy of window size and thresholds. However, study was able to only detect an attack and no mitigation procedures was proposed [25].

A Good feature is important which can be derived using various methods for training the data and classification purposes. Ganapathy et al. [26] specified a method wherein they used an intelligent rule-based attribute selection algorithm to determine the feature set. Seo et al. [27] propose another way of determining the feature set using a clustering-based method.

III. Design and Implementation

SDN features can help defending DDoS attacks. The SDN Switches can act as a firewall when deployed at the edge of a network. With the help of control plain, one could change the route of the packet dynamically and balance the load when an attack occurs. Hence, we proposed an architecture of combining Big Data and SDN as shown in figure 2. The proposed architecture comprises of two main components: SDN and Detection Engine (DE), which are described in subsequent sub sections; and has 3 layers: the infrastructure layer, control layer, and application layer. The infrastructure layer consists of switches/routers, and servers. The switching or routing devices transfer data packets to the next hop in accordance with forwarding rules stored in local memory. The servers can store the big data and run the tasks. The control layer consists of SDN controller which can control the data flow and helps the switch in decision making. Controller runs on a dedicated or distributed server(s), however, logically centralized. It communicates with switches using an OpenFlow API. The application layer consists of Detection Engine (DE) i.e. built using set of big data and networking applications run top of the control layer.

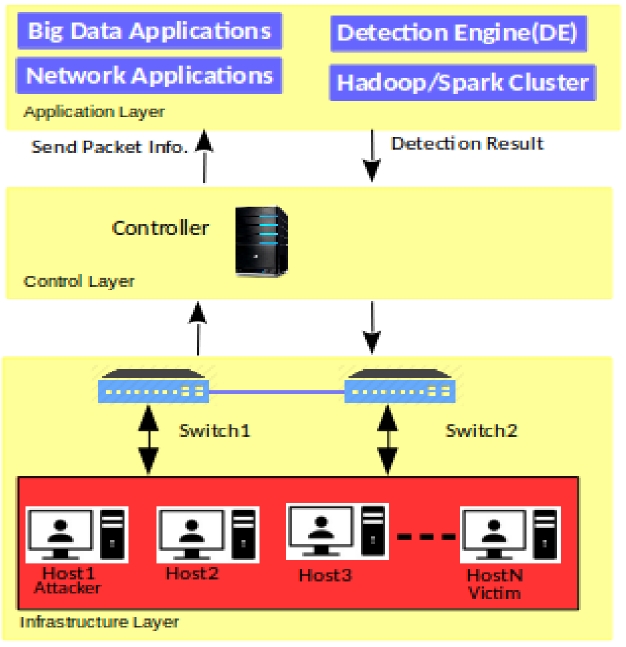


Figure 2: Proposed System Architecture

A. *Software Defined Network*

The proposed system requires the underlying network to be a SDN. The SDN controller plays an important role in the DDoS attack detection and mitigation process. Whenever an incoming packet reaches the switches of the SDN, switch requests the controller for directions on what to do with the packet. Then the controller extracts packet information from all the packets it gets and streams it to the DE. The DE receives packet information and evaluates each packet against the classification model built and gives a decision on whether a particular packet is malicious or not. It relays this information back to the controller. Based on the decision of the DE, the controller sends appropriate flow commands to the switch. Then switch could either forwards the packet on route to its destination or drops it depending on the flow command received. The ability of SDN controller to reprogram the switches dynamically allows us to implement countering mechanisms in real-time.

B. Detection Engine

The Detection Engine (DE) is the heart of our system as it performs the main task of DDoS attack detection. It is a cluster of commodity machines running Apache Hadoop [12] and Apache Spark [13] which communicates with the OpenFlow SDN controller. The DE accepts the real-time network packet's information sent by the OpenFlow controller, feeds this to the pre-trained classification model and relays the result back to the controller. The DE performs this computation for every packet information that the controller sends to it. Since the DE has to perform high computation, we deployed the DE on a cluster of machines. To do this, we used Apache Spark as it does in-memory computing and is 100 times faster than its batch processing counterpart Apache Hadoop. In addition to this, Apache Spark has support for streaming data which helps us stream packet information from the controller to the DE. The Machine Learning libraries (MLlib) which are part of Spark provides implementations of various machine learning algorithms that can be in detection of attacks.

C. DDoS Attack Detection

a. Support Vector Machine

We adopted a single class Support Vector Machine (SVM) which is a type of learning model used for high complex data classification and regression analysis [28]. In SVM, data points are represented as points in space in such a way that points from different categories are separated by a plane. The boundary that gives the largest distance to the nearest observation is called the optimal hyperplane. The optimal hyperplane ensures the fit and robustness of the model. First, data is modeled and the algorithm is trained. Then when new data are encountered their position relative to the "normal" data from training can be used to determine whether it is "out of class" or not - in other words, whether it is unusual or not.

b. Threshold based MapReduce Algorithm

We proposed a threshold based MapReduce (TMR) attack detection method for DE. MapReduce [14] is a programming model for large scale data (Big Data) processing which is the core component of Apache Hadoop [12]. The job of analyzing network log is abstracted as a MapReduce job which can be executed over multiple nodes in a cluster. The Map function process data assigned the master node and produce key-value pairs on individual node. Based on key, value pairs that are generated by map function processes, Reduce function aggregates and returns the result to master node. Finally, master node exports analysis result into respective attack's HBase table.

Algorithm 1 Support Vector Machine

```
 1: function PREDICT($features, w, b$)
 2:     $classification \leftarrow$ sign(dot($features, w$)) $+ b$)
 3:     return $classification$
 4: end function
 5: function TRAIN($Data$)
 6:     $optimized\_sv \leftarrow$ dict()
 7:     for $y_i$ in $data$ do
 8:         for $featureset$ in $data[y_i]$ do
 9:             for $feature$ in $featureset$ do
10:                 all_data.append($feature$)
11:             end for
12:         end for
13:     end for
14:     $all\_data \leftarrow$ None
15:     $step\_size \leftarrow$ [max($all\_data$) $* 0.1$]
16:     $b\_range\_multiple, b\_multiple \leftarrow 2, 5$
17:     $latest\_optimum \leftarrow$ max($all\_data$) $* 10$
18:     $range\_value \leftarrow$ max($all\_data$) $* b\_range\_multiple$
19:     for $step$ in $step\_size$ do
20:         $w \leftarrow [latest\_optimum, latest\_optimum]$
21:         $optimized \leftarrow$ False
22:         while not $optimized$ do
23:             for $b$ in range($-range\_value, range\_value, step * b\_multiple$) do
24:                 for $transformation$ in $transforms$ do
25:                     $w\_t \leftarrow w * transformation$
26:                     $found\_option \leftarrow$ True
27:                     for $i$ in $data$ do
28:                         for $x_i$ in $data[i]$ do
29:                             $y_i \leftarrow I$
30:                             if not $y_i * (\text{dot}(w_t, x_i) + b \geq 1)$ then
31:                                 $found\_option \leftarrow$ False
32:                             end if
33:                         end for
34:                     end for
35:                     if $found\_option$ then
36:                         normalize($optimized\_sv[w\_t]$) $\leftarrow [w_t, b]$
37:                     end if
38:                 end for
39:             end for
40:             if $w[0] < 0$ then
41:                 $optimized \leftarrow$ True
42:             else
43:                 $w \leftarrow w - step$
44:             end if
45:         end while
46:         $norms \leftarrow$ sort($optimized\_sv$)
47:         $w \leftarrow optimized\_sv[norms[0]][0]$
48:         $b \leftarrow optimized\_sv[norms[0]][1]$
49:         $final\_optimum \leftarrow optimized\_sv[norms[0]][0]$
50:     end for
51: end function
```

Apache HBase[15] is a distributed column oriented database built on top of the HDFS. HBase is mainly used when random, real-time read/write access is needed for Big Data. The figure 3 illustrates the MapReduce based packet analysis process.

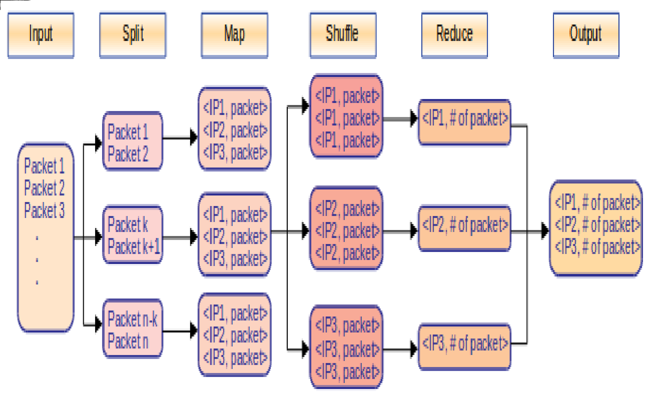


Figure 3: MapReduce Packet Analysis

The state of the art studies have not defined any appropriate method to fix the threshold value for attack detection. However, the threshold value could be determined based on the volume, complexity and damage rate of incoming traffic and the capacity of the distributed network infrastructure by the network administrator.

Algorithm 2 presents the MapReduce process of TCP flooding attack. Similarly, map and reduce functions are implemented for ICMP, UDP and IP flooding attacks. The only difference is in request type parameter value.

Algorithm 2 Threshold Based MapReduce Algorithm

```
1: function MAP($time\_stamp, hbase\_rows$)
2:     $request\_type = hbase\_row[protocol\_type]$
3:     $dest\_ip = hbase\_row[dest]$
4:     if $request\_type =' 6'$ then
5:         $context.write(dest\_ip, 1)$
6:     end if
7: end function
8: function REDUCE($dest\_ip, count[]$)
9:     $total \leftarrow 0$
10:    for $count$ in $counts$ do
11:        $count = total + count$
12:        $context.write(dest\_ip, total)$
13:    end for
14: end function
15: function ANALYZE($dest\_ip, total$)
16:    if $total > Thresold$ then
17:        $hbase.put(dest\_ip, TCPFloodAttack)$
18:    else
19:        $hbase.put(dest\_ip, Normal)$
20:    end if
21: end function
```

c. Fuzzy Deterministic Clustering

We proposed another attack detection method for detection engine known as fuzzy based deterministic clustering (FDC). Fuzzy logic principles can be used to cluster multidimensional data, assigning each point a membership in each cluster center from 0 to 100 percent [28] [29]. This method assigns fuzzy values to data points, such that a single data point can belong to more than one cluster. The total number of clusters and its respective centroid values are generated. The centroid values are used to generate membership values for each data point. Then centroid values are recalculated and steps repeated until these do not change. A threshold membership value for the attack cluster is decided and compared against all data points during the detection phase by DE. The percentage of data points belonging to the attack cluster is calculated and compared with a threshold, if found greater than the DE can be predicts and notify that DDoS attack has risen.

IV. EXPERIMENTATION AND RESULTS

The experimental setup consists of SDN network simulated using Mininet [30]. The SDN and Big Data based tools used in our experiments are as given in table 1. We carried out a number of experiments on the experimental setup. The real-time traffic was ingested into SDN to generate normal and attack traffic datasets using Tshark. TShark is a network protocol analyzer, lets you capture packet data from a live network. The attack scripts developed using Scapy were used to launch TCP, ICMP, UDP flood attacks from the attacker machine in SDN. Scapy is python API for fast packet design and generation. The threshold based MapReduce detection algorithm uses the real-time network logs that are parsed and dumped into HBase table and then analyzed as a MapReduce job over the 3-nodes Hadoop cluster. In order to test this approach, 2000 DARPA Intrusion Detection Scenario-Specific Data Set from MIT Lincoln Laboratory was chosen to experiment with, because this data set is specifically generated for validating DDoS detection methods. The fields extracted from the real-time network log and used by detection methods are shown in table 2. POX is a Python-based controller which plays an important role in defending attacks; reserved for experimentation in the future.

Algorithm 3 Fuzzy Deterministic Clustering

```
1: function DETERMINE_OPCLUSTERS(train_data)
2:     Call CALC_POTENTIAL
3:     $train\_data[Potential\_gap] = pd.Series()$
4:     $train\_data[Potential\_gap] = train\_data[Potential] < 1$
5:     if $train\_data[Potential\_gap]$ then
6:         $train\_data = train\_data.drop(Potential\_gap, 1)$
7:         return
8:     end if
9:     while True do
10:        $return\_val$ = REARRANGE_POTENTIAL
11:    end while
12: end function
13: function CALC_MSHIP(clstr_count, centroids, dataset)
14:     $meship[i,j] \leftarrow \frac{\frac{-2}{d_{ij}^{m-1}}}{\Sigma_{i=1}^{c} d_{ij}^{m-1}}$
            where,
            $m = Fuzzy\ coefficient\ (default\ 2)$
            $c = cluster\ count$
            $i = centroid\ point, j = data\ point$
            $d = euclidean\ distance\ between\ i\ and\ j$
15:     return mship
16: end function
17: function CALC_CENTROID(clstr_count, centroids, dataset)
18:     $centroid_i \leftarrow \frac{\Sigma_{j=1}^{n} u_{ij}^{m} X_{ij}}{\Sigma_{n}^{j=1} u_{ij}^{m}}$
            where,
            $m = Fuzzifier(default\ 2)$
            $c = cluster\ count$
            $u = mship$
            $i = centroid\ point, j = data\ point$
            $d = euclidean\ distance\ between\ i\ and\ j$
19:     $centroids = [centroid_1, \ldots, centroid_c]$
20:     return centroids
21: end function
22: function FUZZY(centroids, train_dataset)
23:     Let normal_centroids[], attack_centroids[] be 1D lists and,
            mship be a 2D list of order nxc;
            where,
            n = number of data points,
            c = number of centroids;
24:     mship_df is a df represent the 2D mship list
25:     $train\_data \leftarrow dataframe\ for\ training\ data$
26:     $mship \leftarrow$ CALC_MSHIP(clstr_count, centroids, train_data)
27:     $centroid \leftarrow$ CALC_CENTROID(clstr_count, centroids, train_data)
28:     while True do
29:         $old\_mship \leftarrow mship$
30:         $old\_mship\_df \leftarrow mship\_df$
31:         $mship \leftarrow$ CALC_MSHIP(clstr_count, centroids, train_data)
32:         if $old\_mship == mship$ then
33:             break
34:         end if
35:     end while
36:     Identify attack clusters.
37:     $min\_mship \leftarrow$ minimum membership in attack clutser.
38:     while True do
39:         Capture packets during runtime.
40:         $test\_data \leftarrow dataframe\ for\ test\ data$
41:         $mship \leftarrow$ CALC_MSHIP((clstr_count, centroids, test_data)
42:         $tot\_attack\_packets \leftarrow$ Total attack packets
43:         if $tot\_attack\_packets \geq threshold * total\_packets$ then
44:             return DDoS Attack
45:         end if
46:     end while
47: end function
```

Big Data Tools	Apache Hadoop, Apache Spark, HBase, Flume
Cluster Size	1 master & 2 slaves
Language(s)	Python & Java
Controller	Pox
Network Emulator	Mininet
Attack Traffic Generation	Scapy

Table 1: Experimentation Environment

Executing a HBase command scan 'DDoSLog', will produce the result, shown in figure 4. And the analysis result of MapReduce approach will produce the result, shown in figure 5. Similarly, UDP, HTTP, and ICMP packets can be analyzed from the network log.

Pckt. Interval	**Src. IP**	**Dest. IP**	**Protocol**
0.000100000	10.0.0.1	10.0.0.5	1
0.000210000	10.0.0.1	10.0.0.5	6

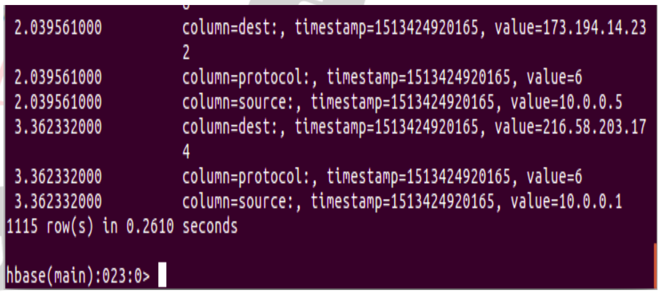
Table 2: Dataset Fields

Exp. #	Traffic Rate (pkts/s)	SVM detection ratio
Exp.1	Normal traffic(x)	75%
Exp.2	50x	82%
Exp.3	100x	95%

Table 3: SVM attack detection ratio

Exp. #	Traffic Rate (pkts/s)	FDC detection ratio
Exp.1	Normal traffic(x)	75%
Exp.2	50x	82%
Exp.3	100x	95%

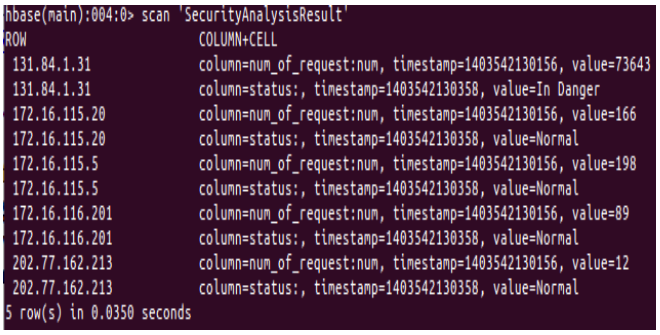
Table 4: FDC attack detection ratio



```
 2.039561000          column=dest:, timestamp=1513424920165, value=173.194.14.23
                      2
 2.039561000          column=protocol:, timestamp=1513424920165, value=6
 2.039561000          column=source:, timestamp=1513424920165, value=10.0.0.5
 3.362332000          column=dest:, timestamp=1513424920165, value=216.58.203.17
                      4
 3.362332000          column=protocol:, timestamp=1513424920165, value=6
 3.362332000          column=source:, timestamp=1513424920165, value=10.0.0.1
1115 row(s) in 0.2610 seconds

hbase(main):023:0>
```

Figure 4: Parsed DDoS Log



```
hbase(main):004:0> scan 'SecurityAnalysisResult'
ROW                    COLUMN+CELL
 131.84.1.31           column=num_of_request:num, timestamp=1403542130156, value=73643
 131.84.1.31           column=status:, timestamp=1403542130358, value=In Danger
 172.16.115.20         column=num_of_request:num, timestamp=1403542130156, value=166
 172.16.115.20         column=status:, timestamp=1403542130358, value=Normal
 172.16.115.5          column=num_of_request:num, timestamp=1403542130156, value=198
 172.16.115.5          column=status:, timestamp=1403542130358, value=Normal
 172.16.116.201        column=num_of_request:num, timestamp=1403542130156, value=89
 172.16.116.201        column=status:, timestamp=1403542130358, value=Normal
 202.77.162.213        column=num_of_request:num, timestamp=1403542130156, value=12
 202.77.162.213        column=status:, timestamp=1403542130358, value=Normal
5 row(s) in 0.0350 seconds
```

Figure 5: TCP Packet Analysis Result

As can be seen from the table 3 and 4, SVM provides an error rate of more than 5% as compared to FDC which exhibits an error of 2%. The study considered the normal traffic rate of 500 packets/sec. The figure 6 shows the time comparison of three methods. The accuracy results of 3 experiments are plotted in figure 7. The Receiver Operating Characteristic (ROC) summarizes the model's performance by evaluating the tradeoffs between true positive rate (sensitivity) and false positive rate (1-specificity). As can be seen from this figure and the argument put forth, the MapReduce detection method provided a high detection rate of 100% as compared to other methods which have detection rate lower than this value. The main cause of getting high accuracy of MapReduce detection method is that dumping real time network log data into HBase table and using that for analysis.

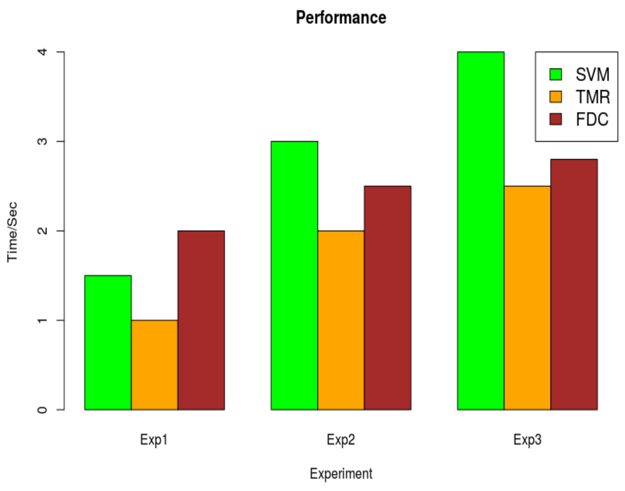


Figure 6: Experiments Vs Time

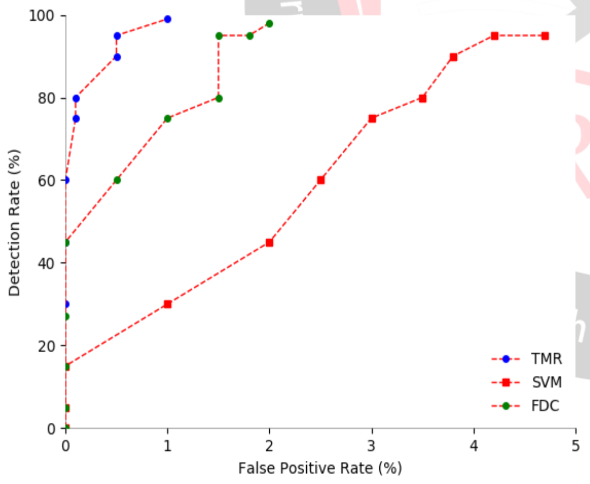


Figure 7: ROC Curve

V. Conclusion and Future Scope

This research proposed a real-time solution for detecting DDoS attacks in SDN environment and also compared the effectiveness of proposed methods. The proposed system could further be enhanced to detect and counter other types of DDoS attacks. Robustness and security can be further improved by making the controller resistant to DDoS attacks.

References

[1] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms", SIGCOMM Comput. Commun. Rev., vol. 34, no. 2, pp. 3953, Apr. 2004.

[2] P. Morreale and J. Anderson, "Software Defined Networking: Design and Deployment", Taylor & Francis, 2014.

[3] "Software-Defined Networking: The New Norm for Networks and Open Networking Foundation" - Open Networking Foundation - ONF White Paper April 13, 2012.

[4] T. Kitten, DDoS Attacks Against Banks Increasing, blog, BankInfoSecurity, 24 Aug. 2015; www.bankinfosecurity.com/ddos-a-8497.

[5] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig, "Software-Defined Networking: A Comprehensive Survey", 8 Oct 2014.

[6] Cisco's VNI Global Traffic Forecast, 2015-2020

[7] KrebsOnSecurity Hit with Record DDoS, blog, 21 Sept. 2016.

[8] US Computer Emergency Readiness Team, Heightened DDoS Threat Posed by Mirai and Other Botnets, alert TA16-288A, 14 Oct. 2016 (revised 30 Nov. 2016); www.us-cert.gov/ncas/alerts/TA16-288A.

[9] K. Bakshi, Considerations for Big Data: Architecture and Approaches, In: Proceedings of the IEEE Aerospace Conference, pp.17, 2012.

[10] Sachchidanand Singh and Nirmala Singh, "Big Data Analytics", International Conference on Communication, Information Computing Technology (ICCICT), Oct. 2012.

[11] Steve Hoffman, "Apache Flume: Distributed Log Collection for Hadoop", Packt Publishing, 2013.

[12] Tom White, "Hadoop: The Definitive Guide", O'Reilly Media, 2012.

[13] Michael J Franklin Scott Shenker Matei Zaharia, Mosharaf Chowdhury and Ion Stoica, "Spark: Cluster computing with working sets", In USENIX conference on Hot topics in cloud computing, pp. 10-16, 2010.

[14] J.Dean and S.Ghemawat, "MapReduce: Simplified data processing on large clusters", Hindawi Publishing Corporation Advances in Multimedia, vol. 51, no. 1, pp. 107-113, 2008.

[15] Lars, "HBase: The Definitive Guide", O'Reilly, 2011.

[16] K. Chodorow, MongoDB: The Definitive Guide, O'Reilly, 2nd edition, 2013.

[17] Eben Hewitt, "Cassandra: The Definitive Guide", O'Reilly, 2010.

[18] Zubair Nabi, "Pro Spark Streaming: The Zen of Real-Time Analytics Using Apache Spark", 2016.

[19] Van d V J S, Van D W B, Lazovik E, et al. "Dynamically Scaling Apache Storm for the Analysis of Streaming Data". Proceedings of IEEE First International Conference on Big Data Computing Service and Applications. IEEE Computer Society, pp. 154-161, 2015:

[20] Pratyusa K Manadhata Alvaro A Cardenas and Sreeranga P Rajan, "Big data analytics for security", IEEE Security and Privacy, vol. 6, pp. 74-76, 2013.

[21] Jerome Francois, Shaonan Wang, Walter Bronzi, R State, and Thomas Engel. "Botcloud: Detecting botnets using MapReduce". In Information Forensics and Security (WIFS), 2011 IEEE International Workshop, pages 1-6, 2011.

[22] Yeonhee Lee and Youngseok Lee, "Detecting DDoS attacks with Hadoop", In Proceedings of The ACM CoNEXT Student Workshop, pp. 1-2, 2011.

[23] Johan Mazel Romain Fontugne and Kensuke Fukuda, "Hashdoop: A MapReduce framework for network anomaly detection", IEEE Conference (INFOCOM WKSHPS), pp. 494-499, 2014.

[24] Sufian Hameed, Usman Ali ,"On the Efficacy of Live DDoS Detection with Hadoop", 2015

[25] S. M. Mousavi, "Early Detection of DDoS Attacks in Software Defined Networks Controller", Ph.D. dissertation, Carleton University, 2014.

[26] S. Ganapathy, K. Kulothungan, S. Muthuraj kumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: a survey", EURASIP Journal on Wireless Communications and Networking, vol. 1, 2013.

[27] J. Seo, J. Kim, J. Moon, B. J. Kang, and E. G. In, "Clustering based Feature Selection for Internet Attack Defense", International Journal of Future Generation Communication and Networking, vol. 1, pp. 91-98, 2008.

[28] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, "An Introduction to Statistical Learning".

[29] Jang, J.-S.R.; Sun, C. T, Mizutani E., "Neuro-Fuzzy and Soft Computing", Prentice Hall Inc., USA, 1997.

[30] Rogrio Leo Santos de Oliveira, Ailton Akira Shinoda, Christiane Marie Schweitzer, Ligia Rodrigues Prete, "Using Mininet for Emulation and Prototyping Software-Defined Networks", Brazil, IEEE, 2014.