Real-Time Attacks Detection Model And Platform Using Big Data And Machine Learning

Mostafa Mohamed Shendi, Hatem Mohamed Elkadi , Mohamed Helmy Khafagy

Abstract: one of the most serious attacks nowadays is the Distributed Denial of service (DDoS) attack. DDoS attacks can be of two types: the layer-three (network layer) attacks and layer-seven (application layer) attacks, which can potentially lead to cyber-attack resulting in financial and reputational losses. Hence, network analytics play an important role in protecting the security of organizations. Traditional data analysis models have difficulties in defeating these attacks since they consume too much time analyzing different logs from different devices at the same time. Big Data analytics plays a major role in analyzing and correlating large volumes of disparate and complex data from different sources in different formats. Hence, we propose a model that combines Big Data and machine learning to proactively detect DDOS attacks through analysis, detection, and classification of network traffic. By using this model, organizations could gain higher service security and availability. The experimental results show that different attack scenarios are well classified, and DDoS attacks could be detected in the early stages

Real-time: Big data processing, anomaly detection, Machine learning, Hadoop, Spark, Kafka, DDOS, Log Files, Log Analysis.

INTRODUCTION

NETWORK security, as the basis of internet security, has become of utmost importance in all areas of industry and business, including financial institutions, service providers, etc. Recently, infrastructure, web applications, and network services have suffered from intruder attacks. Hackers are continuously trying to compromise systems through new types of Distributed Denial of Service (DDoS), which affect the application layer as well as the network layer. Attackers try to deny access to web services and slow down access to network resources by flooding the network, server, or application with fake traffic. A Distributed Denial of Service (DDoS) attack is a malicious attempt to make the victim machine or web server and network resources unavailable to legitimate users by disrupting the services and gaining malicious control into a large number of computers, compromised machines, and bots, called slaves or zombies. Then the attackers move on to instruct the zombies in The network to execute attacks at a certain time, such as running malware to generate a large set of attack packets and sending a large number of requests to a target server. Therefore, it would be unable to respond to any of the requests [1]. Attackers are always looking for new vulnerabilities and methods to exploit the protection solutions and turn the system to be overloaded. Banks and financial institutions, service providers, and social applications experience a high share of DDOS attacks in recent years [2]. Table 1 highlights some types of DDoS attacks mentioned in [3] driven by different techniques, targeted for various resources, and exploiting their vulnerabilities.

***Table1** Different types of DoS attacks [3]*

Types of DDOS	Targets	Vulnerabilities
Attack on network devices	Network devices/hardware such as routers, switches, and firewalls	Vulnerability/bug in the device's software
Application attack	Application services	Application limitations and weaknesses
Data flooding	Network bandwidth or capacity of the networks or the servers	Limited network bandwidth/Limited server capacity of processing requests
Protocol attack	Protocol services	Protocol limitations and weaknesses such as ARP poisoning, IP spoofing, and TCP SYN flooding
Operating system attack	OS services	Vulnerability/bug in OS software

Different types of DDoS attacks can be applied to OSI different layers in Figure 1. In this work, we describe the testing of the most recent attacks on the application layer and the network layer [4].

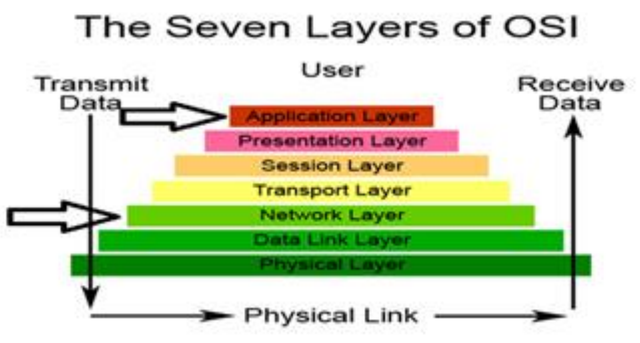


***Fig.**1 "OSI seven layers."*

The huge amount of data generated in real-time affects the performance of network analysis in terms of scaling the increasing volume of data. Therefore, it is important to produce network security analytics performance reports in the real and near real-time and not analyze each device separately. Analyzing this huge amount of network traffic at once by traditional ways is inefficient, costly, and may lead to systems slowness and the intrusion detection itself to fall [5]. This issue has created new challenges and motivated researchers to find new methods to defend such attacks and proactively detect anomalous behavior. In particular, anomaly detection leads to early detection of irrelevant patterns or up normal events in the network. It monitors major network ratios in real-time and sends an alarm if any anomalous actions are detected in the network. The detection of such attacks plays an important role in maintaining the security of networks. Here, we introduce a method for proactive detection of DDOS attacks, by classifying the network status and monitoring the servers and network devices utilization in the detection stage of the proposed framework. There are several aspects that we need to check on the network's designs and its components, such as firewalls, routers, switches, and backend servers. The customers of the network include internal and external users.

The Need for Big Data and Machine Learning

Hence, the proposed framework is the architecture of combining Big Data and machine learning techniques to detect and classify any anomalous behavior to achieve fast and early detection of DDOS attacks. Big data analytics plays an important role as the network logs are extremely large and updated frequently. Using Bigdata Analytics enables the analysis of large volumes of disparate logs and integrating them from different sources in different formats in real-time. Bigdata analytics covers ingesting, storing, and analyzing the logs to discover hidden patterns and violations of insight. The contribution of Bigdata includes depth, time, storage, processing, analysis, and scalability. Even though Machine learning techniques have been adopted to detect and classify network traffic based on some features such as (average packet size, interval time, packet rate, packet size, bit rate, etc.) that are used to measure and classify the network traffic as normal or as a type of DDOS, DDOS has a common pattern like the following: traffic has same average packet size, increasing in the attacked packed rather than a normal packet; the interval time is too small to consume the resources faster, packets have a high bit rate for network attacks [6], attackers targets consuming resources and make the service unavailable to end-users. ML-based DDOS detection approaches are either supervised or unsupervised. Supervised ML approach for DDOS detection relies on the availability of labeled network traffic datasets, whereas unsupervised ML approaches detect attacks by analyzing the incoming network traffic. Both are challenged by a large amount of traffic data, low detection accuracy, and high false-positive rates. In this paper, we present a model that consists of big data analytics using Kafka, Spark, Hive and machine learning box for DDoS detection based on logs analysis, with the integration of SPARK we were able to apply this on a large scale of data and enable it in real-time. The supervised part of the approach allows us to reduce the irrelevant normal traffic data for DDoS detection. At the same time, the Bigdata model allows us to classify the DDoS traffic by reducing false-positive rates accurately.

DDOS ON THE NETWORK LAYER

Network layer attacks are the attacks that exploit the vulnerabilities of the network and transport layer, and its protocols Figure 2. Network layer Attacks exploit the bandwidth of the connection. They are performed by spoofing the IP address or blocking the ports by sending large amounts of malicious requests. Nowadays, most of the network layer attacks can be prevented by the latest firewall systems and IDS/IPS systems. The protocols mostly used to perform these types of attacks are Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) [6].

Types of Network layer DDoS attacks:

- UDP flood: the attacker sends a large number of UDP packets to random ports on the target server.
- ICMP flood: the attacker sends packets without waiting for replies, result in consuming both outgoing and incoming bandwidth of the victim's network.
- Smurf: the victim is flooded with ICMP echo-reply packets. The packets are sent at broadcast addresses; this creates a large amount of echo-response packets, thereby making the network unstable and causing network congestion to the victim.
- Ping of death: victim's system is flooded with many malformed pings uses oversized packets by ping command. Causes memory buffers overflow allocated for the packet.

DDOS on application layer

Application Layer Attacks targets the protocols of the application layer Figure 2. These attacks are very hard to detect as they use a legitimate TCP connection. Also, these attacks don't exhaust the network bandwidth but exhaust server resources like CPU cycles, memory, or socket connections [7]. Application layer DDoS attacks are designed to attack the application itself, focusing on specific vulnerabilities resulting in the application not being able to deliver content to the user. Application layer attacks are designed to attack specific applications; the most common are web servers. Such attacks are usually low-to-mid volume since they have to adapt to the application protocols such as protocol handshakes and protocol compliance. With this traffic, an attacker can fully utilize bandwidth and server resources until one (or both) is crashed and can no longer handle the requests. The server crashes or has no enough resources to allow legitimate customers to access your web service. If the normal application traffic is 600 connections at a time throughout the day and server runs normally, then 600 clients will probably be able to connect without affecting the servers. However, with a DDoS attack, it will be thousands of connections from numerous different IPs at one time. If the server can't handle this number of connections at a time, then it could be vulnerable under a DDOS attack.

Types of Application layer DDoS attacks:

- Session Flooding: The attacker sends a session request with a huge volume more than a normal user.
- Request Flooding: The attacker sends a larger number of requests than a normal user.
- Asymmetric attack: The attacker makes requests at high workloads like downloading big files or responding to a database intensive request.

Challenges of Application Layer DDOS detection:

- Most security devices fail to detect application-layer attacks because the attacker uses normal HTTP Packets
- Application layer DDoS targets one of the server resources, e.g. (CPU, database, memory, socket connection), which do not affect the functionality of other resources. Hence, with less traffic and bandwidth, a server can be brought down. Most detection systems use traffic volume to detect an attack; hereby, application-layer attacks detection fails with current detection techniques.
- Application layer DDoS attacks are often mistaken for flash crowds. Flash crowds are a sudden increase in the amount of traffic on a service whereby we have to segregate legitimate flash crowd data and illegitimate application-layer DDoS attacks.

MODEL DESIGN AND IMPLEMENTATION

We are proposing a model [figure 2] that consists of big data techniques and Machine learning models to detect and provided with a detailed report on each graphic within the web applet, as well as by email. For more information on using the Graphics Checker Tool Or any other graphics related topic, contact the IEEE Graphics Help Desk by email at graphics@ieee.org. Classify the DDOS attacks, and for future work, this model will be able to detect more types of attacks after identifying attack patterns and train the model to detect it. Our model is designed as below:

- Collect log files from different devices.
- Transform, collect, aggregate, and stream the data.
- Process the collected data in real-time and batch jobs.
- Store the processed data in the NoSQL database to save the results and create historical reports.
- Analyze these events, trigger real-time alerts and apply machine learning
- The classifier will be trained in a supervised mode to classify the traffic.

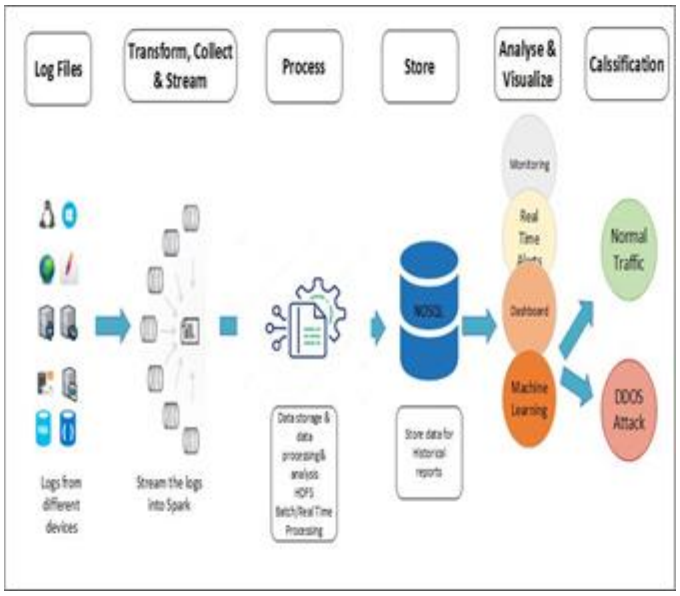

Log Files
Transform, Collect & Stream
Process
Store
Analyse & Visualize
Calssification
Monitoring
Real Time Alerts
Dashboard
Machine Learning
Normal Traffic
DDOS Attack
NOSQL
Logs from different devices
Stream the logs into Spark
Data storage & data processing& analysis HDFS Batch/Real Time Processing
Store data for Historical reports

Fig.2 *Design of attack detection model*

The model captures the logs from different sources [figure 3]. (e.g., Web application log, Operating system, firewall logs, and Pinger log). In real-time, streams the logs content to the Spark through Kafka topics, checks the attacks patterns on the Spark and saved on HIVE, generates real-time dashboard, and alarming reports for the current data streaming and uses apache zeppelin for the historical dashboard.

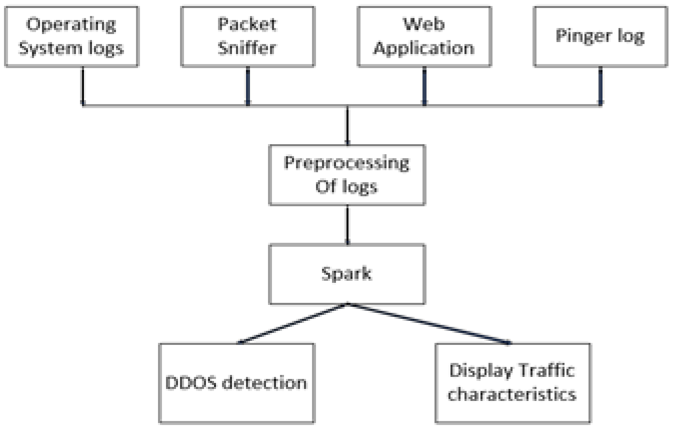


Fig.3. *Capture and stream data*

To classify the network traffic, we use Bigdata tools and machine learning techniques, which was explained previously. Firstly, we collect the logs from the firewall, operating system, and Pinger then we process the logs information on the Spark in real-time. Finally, we apply machine learning classifiers to classify the traffic type. Hence, the model will support the recognition of the current network status and early detection of DDoS attacks. The details of the detection of DDoS attacks are shown in

Figure 4 and following the below steps:

- Collect the network devices logs (firewall, Pinger), Operation systems (CPU- memory utilization), and web application logs.
- Ingest and process the logs to analyze data through Kafka.
- Kafka – pub-sub messaging system, fast, scalable, and topic-oriented.
- Processing Kafka HDFS (Storage) & Spark for real-time events alert and analyze.
- Store the analysis result on HIVE- a data-warehousing framework built on top of Hadoop.
- Create dashboards for the traffics from historical data using apache zeppelin.
- Train ML Model to classify events as DDOS or legitimate.

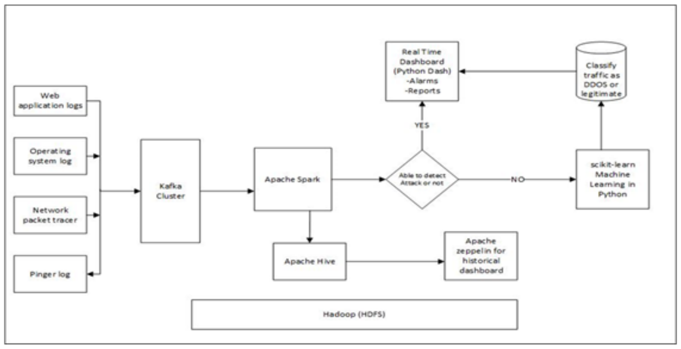

Web application logs
Operating system log
Network packet tracer
Pinger log
Kafka Cluster
Apache Spark
Able to detect Attack or not
YES
NO
Real Time Dashboard (Python Dash) -Alarms -Reports
Classify traffic as DDOS or legitimate
scikit-learn Machine Learning in Python
Apache Hive
Apache zeppelin for historical dashboard
Hadoop (HDFS)

***Fig.4**. The proposed model Architecture*

Spark output has two scenarios, the first is where Spark can verify the traffic connection as an attack and detect the used devices IPs, and thus it will be stored as signature-based attacks and will use it on training our model. The second scenario is passing the traffic to the machine learning model to classify it as an attack or normal, then goes through the trained classifier and returns the classification result to the spark stream figure 5. Network traffic monitoring might include much more information about the overall system behavior than log files or page-tags information output. But it is also more complicated; here we are covering the network side and the application logs.

***Fig.5**. Machine learning stage*

Logs Acquisition and Arrangement using Bigdata Techniques

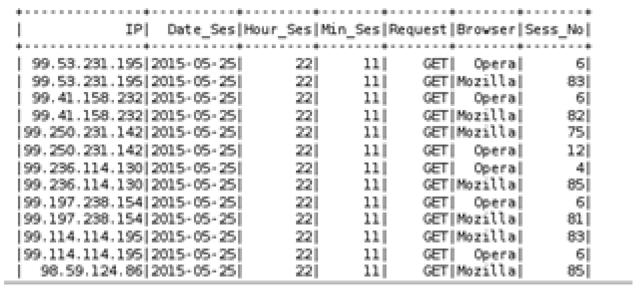
The most effective way to mitigate a DDoS attack is to be able to detect it immediately when it begins. Several clues indicate an ongoing DDoS attack is taking place, such as An IP address makes x requests over y seconds (high number of connections in the small interval) Number of connections versus the average normal number of your application connections (per hour) The TTL (time to live) on a ping request takes much time, or it goes time outThe Packet size, and type of transport protocol. We will monitor the above clues as it will help on detecting the DDOS attacks on early-stage and send notification alerts to the administration team Figure 5 shows the flow of traffic events in the Bigdata Model DDOS detection system. A packet sniffer, Application log, Pinger log, and Operating system is used to capture traffic then sent it to spark. The information is preprocessed to extract the required information needed for the detection. We will monitor the above clues as it will help on detecting the DDOS attacks on early-stage and send notification alerts to the administration team Figure 5 shows the flow of traffic events in the Bigdata Model DDOS detection system. A packet sniffer, Application log, Pinger log, and Operating system is used to capture traffic then sent it to spark. The information is preprocessed to extract the required information needed for the detection.

Check Server CPU/Memory for DDOS Attack

The DDOS attacks are designed to overload server resources, use up all available connection/bandwidth/throughput. First, to do is use TOP commands, which will provide us with the server's load and the highly utilized services. Therefore, the model traces the server CPU and memory utilization and correlate with the application logs to detect DDOS behavior. We have developed a shell script to continuously monitor CPU usage and MEMORY usage of the application process and write /convert this data to CSV.

CHEK GET application URL (ping request)

One of our measures to check the attacks on the application is to check the TTL (Time to Live) on a ping request if it takes much time or returns a time out. The term time-to-live is also used to describe the time for which a DNS record can be returned from the cache. We have developed a ping script as a function get_data($url), run the job from a scheduler; we schedule it from crontab to run each five minutes then we have ingested the pinger.log to our model and check the application response since DDOS attacks wats away the bandwidth, the ping time will be too long or will return a time out. Will correlate any slowness with other events. Check Web application logSpooling the web application log through Kafka to SPARK, we query for the number of connected sessions from the same IPs addresses in a small interval of time. Hence it is not human behavior. Figure 6 we list the number of sessions requested from the same IPs, browser used request type, and the connection time interval.



```
+--------------+----------+--------+-------+-------+-------+-------+
|            IP|  Date_Ses|Hour_Ses|Min_Ses|Request|Browser|Sess_No|
+--------------+----------+--------+-------+-------+-------+-------+
| 99.53.231.195|2015-05-25|      22|     11|    GET|  Opera|      6|
| 99.53.231.195|2015-05-25|      22|     11|    GET|Mozilla|     83|
| 99.41.158.232|2015-05-25|      22|     11|    GET|  Opera|      6|
| 99.41.158.232|2015-05-25|      22|     11|    GET|Mozilla|     82|
|99.250.231.142|2015-05-25|      22|     11|    GET|Mozilla|     75|
|99.250.231.142|2015-05-25|      22|     11|    GET|  Opera|     12|
|99.236.114.130|2015-05-25|      22|     11|    GET|  Opera|      4|
|99.236.114.130|2015-05-25|      22|     11|    GET|Mozilla|     85|
|99.197.238.154|2015-05-25|      22|     11|    GET|  Opera|      6|
|99.197.238.154|2015-05-25|      22|     11|    GET|Mozilla|     81|
|99.114.114.195|2015-05-25|      22|     11|    GET|Mozilla|     83|
|99.114.114.195|2015-05-25|      22|     11|    GET|  Opera|      6|
|  98.59.124.86|2015-05-25|      22|     11|    GET|Mozilla|     85|
```

Fig.*6. count sessions from weblog*

Network packet tracer
A packet sniffer, "Wireshark," [9] is used to capture traffic and stream it through Kafka to send it to spark. The traffic information is preprocessed to keep the required information needed for the detection of anomalies only. Retain the source Ip address, destination IP address, packet size, and type of transport protocol. The model was processing the information and displayed network characteristics that help in detecting the anomalies' behavior in the network.

Machine learning Techniques
Machine learning is a technique of data analysis that automates analytical model building. A supervised learning algorithm takes a known set of input data and known responses to the data (labeled) and trains a model to generate reasonable predictions for the response to new data [10]. The proposed framework is integrating the machine learning algorithms with big data processing[11,12], which helps to simulate attacker's behavior with computational intelligence, increase the percentage of accuracy in detection and also identify traffic patterns and make decisions with minimal human intervention. Machine learning is used to detect and classify DDOS attack based on some features such as (average packet size, interval arrival time, packet rate, packet size, bit rate, etc.) the mentioned values are used to measure whether the traffic is normal or is a type of DDOS. The dataset we used as a case study in this paper (NS2) is contributed by Alkasassbeh et al. [13] and is publicly available for analysis. We selected NS2 due to the consideration of recent kinds of DDoS attacks on the application and network layers, which are reflected by the traffic features in this dataset. (NS2) containing five target classes cover the application and network DDOS attacks: HTTP flood, Smurf, SQL Injection DDOS, and normal. The first four classes are representing the DDOS attacks, while the last class refers to legitimate traffic. NS2 has 27 features, we did some correlation analysis on it and found out that many of the features are strongly correlated and some of the variables can be neglected, DDOS attacks include the below criteria:
- The same average of the packet size
- The number of packets increased in the attack rather than the normal packets number
- The small interval time (to allow attackers to consume resources)
- The attack packets always have a high bit rate (for network attack)

In addition to a linear correlation between 'pkt_size' and 'pkt_avg_size' features.

We have tried many ML models, considering the imbalance of the data we did not only focus on the accuracy but mainly on the macro average to calculate the F1 score.

Data Imbalance
Table 2 shows that (NS2) is a large dataset of more than a million records, out of which 108,927 represent the attack traffic. It indicates that 10.38% of the dataset corresponds to DDoS class while the remaining belongs to a normal class. Hence (NS2) is skewed toward the normal class where the majority of records belong to normal traffic. Class imbalance is a well-known problem in data science, as shown below:

Table 2 *Datasets classes*

Class	Count
Normal	939618
UDP-Flood	97520
Smurf	6211
SIDDOS	3198
HTTP-FLOOD	1997

To overcome the data imbalance issue, we have used class weighting technique, giving a low class a higher weight; thus, the attacks classes will affect the result much more, the below equation to get each class weight:
〖W〗_class = (∑_0^n▒C_i)/(C_class)
Latex: W_{class} = \frac{∑_({0}_i^{n}C)}{C_{class} }
"weight of the new class."
Where Ci is the class samples, the sum of all classes is divided by the class count.
Data Preprocessing
We have used two approaches, the standard scaling and min-max scaling [14], the two formulas are as follows.
The standard scaling allows us to capture the distribution of the data centralized around zero.

Z=(X-μ)/σ
Latex: z = \frac{x-μ}{σ}

The Min-Max Scaler, which Squints the number into a small scale, usually from 0 to 1.
Z= (X-min⁡(X))/(max⁡(X)-min⁡(X))
Latex: Z = \frac{X-min⁡(X) }{max⁡(X)-min⁡(X) }

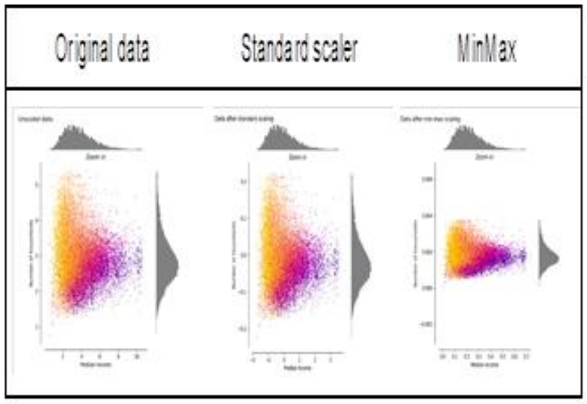


***Fig.7**. preprocessing scaler*

Standard scaler keeps the data distributed but scales the values down, as for the Min-Max scaler it changes the data distribution figure 7, we have tried both on our experiment.
In addition to handling the categorical classes when encoding them, we will give each category a representing number and splitting the data into training and dev set, with a dev set of 10% of the data size.

Application of the proposed models
In this model, the experiments were performed on Ubuntu 13 platform. Dataset is input to a particular machine learning model one-by-one, and then the analysis is made under different parameters for optimal response. Each classifier was trained on our dataset using 90% of the collected data, and 10% were used as test data to measure the classifier's efficiency. We measure the classification accuracy and other metrics on test data to evaluate the performance of the classifiers applied in this work.

Experiment one step classifier
The experimental setup consists of many machine learning models; some achieved high accuracy. We measured the model accuracy on the training and test set to avoid overfitting and used cross-validation to ensure that the results are not an accident and that they are true numbers. We also used multiple measures to assess our models, such as precision, recall, f1-score, with macro averaging, to ensure that the score we got covers the minority and the majority of classes. Table 4 contains information about real and predicted classifications carried out by the classification models. We use Logistic regression, Passive Aggressive, SGD, Random Forest, and Decision Tree classifiers. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 [15].

***Table 4** trained classifiers and results*

Classifier	train ccuracy	train f1 Macro	test f1 Macro	scaling
Logistic Regression (without class weights)	0.979	-	0.38	-
Logistic Regression	0.77	0.66	0.66	standard
Logistic Regression	0.45	0.59	0.59	minmax
Passive Aggressive Classifier	0.007	0.18	0.18	-
Passive Aggressive Classifier	0.67	0.35	0.36	standard
SGD Classifier (eta0:0.1- Lr:adaptive - loss: log - penalty:l1)	0.98	0.80	0.79	minmax
SGDClassifier (eta0:0.1- Lr:adaptive - loss: log - penalty:l2)	0.98	0.82	0.81	standard
Decision Tree Max_depth:3 - min_samples_leaf:10	0.97	0.70	0.71	minmax
Decision Tree Max_depth:3 - min_samples_leaf:10	0.98	0.70	0.70	standard
Random Forest, 500 learner	0.98	0.83	0.82	minmax
Random Forest, 500 learner	0.98	0.84	0.83	standard
Linear SVC	0.98	0.82	0.82	minmax
Linear SVC	0.98	0.82	0.82	standard
GaussianNB	0.96	0.75	0.75	standard

The best classifier
Table 5 shows the best model we had is a Random Forest, which is an ensemble model that is based on the bagging technique. The Random Forest Classifier we used had 500 estimators, with a max depth of 10 and min samples per leaf equal to 10. We also applied the class weights that we came up with early, applied 3-fold cross-validation using Sklearn functionality. We use precision and recall measure [18,19] Here are the detailed results of the best classifier applied on all attacks type:

***Table 5** Random Forest classifier for each attack type*

Also, we had very promising results with the Stochastic Gradient Descent Classifier (SGDClassifier), as it came second after the Random Forest with a macro f1-score of 0.81 on the test data, and 0.82 on training data.
Two-step classifiers

We performed another experiment with two separate classifiers tables 6,7. The first one is to predict whether the traffic is an attack or not, and the second to classify which attacks are the packet. Splitting the one task of predicting the packet is an attack or not and which attack is into two problems and giving the two classifiers small problem scope to work with, but also making the separating a bit harder. In our experiment, we used 108926 samples of the data for the attack-or-normal, and for the which-attack problem, we used 1900 samples for each of the classes.

***Table 6** The first classifier (detects the packets (attacks-or-normal))*

	Precision	recall	F1-score	support
Attack	1.00	0.87	0.93	10963
Normal	0.88	1.00	0.94	10823
Accuracy			0.93	21786
Macro avg	0.94	0.94	0.93	21786
Weighted avg	0.94	0.93	0.93	21786

***Table 7** The second Classifier classify which is the attack*

	Precision	recall	F1-score	support
Attack	1.00	0.87	0.93	10963
Normal	0.88	1.00	0.94	10823
Accuracy			0.93	21786
Macro avg	0.94	0.94	0.93	21786
Weighted avg	0.94	0.93	0.93	21786

Discussions of the Model Result

The experimental setup consists of network attack simulated tools, and we carried out several experiments. We used DDOSIM [16] and BoNeSi [17] to simulate HTTP-GET floods from large-scale networks and generates ICMP, UDP, SIDDONS, and TCP (HTTP) flooding attacks from a defined botnet. Used Tshark to capture packet data from the network, streamed the logs, created Kafka Topic, and build Kafka, producer, using Python to read the logs then build Spark Pipeline to process Stream apache log and write it to Hive. Use the random forest classifier, which is an ensemble learning model that is composed of decision trees. The decision tree uses CART algorithms, split each node based on feature K and threshold tk, then the pair (K, tk) is chosen that achieves the purest split.

The purity of the split can be calculated using the Gini impurity equation

G_(i =1-) ∑_(k-1)^n▒P_(i,k)^2

Where pi,k is the ratio of class k instances among the training instances in the node.

The gini impurity indicates how impure the split is, for example a node with the following number of items: [0, 49, 5], total of 54 gini score is 1-(0/54)2 -(49/54)2 -(5/54)2 = 0.168, and a node with the following items: [0, 53, 1], total of 54 gini score is 1-(0/54)2 - (55/54)2 - (1/54)2 = 0.036.

J(K,t_k)=m_left/m G_left+ m_right/m G_right

The model tries to minimize the following loss function:

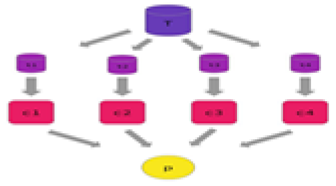


***Fig.8**. Random forest-bagging*

The random forest is an ensemble learning approach that makes use of multiple weak learner "decision tree," using a bagging teaching which is short for bootstrap aggregation, that splits the data with replacement into multiple splits and trains multiple classifiers, each on a separate split, all of the classifiers are decision trees, then at inference, the predictions of all of them are aggregated to make the final prediction [17]. We used max depth of 10 for the trees, minimum samples per leaf as 10, and 500 estimators and kept the default values for the other parameters as a Sklearn library.

Integration with Spark

Integrate the model with the spark stream, and we set the backend of Sklearn to be Spark by using the joblib-spark library [19] provides Apache Spark backend for joblib to distribute tasks on a Spark cluster.
from joblibspark import register_spark
register_spark() # register spark backend
with parallel_backend('spark', n_jobs=1): predections = classifier.predict(packet_data) Sklearn based model will run on a backend of Spark, thus make the model predictions keep up with the spark stream.
The accuracy of our experiment shows that our model is efficient enough for the early detection of DDoS attacks. Classify the traffic in suitable computing time. Table 8 shows the result after applying the model to new data in a controlled simulation.

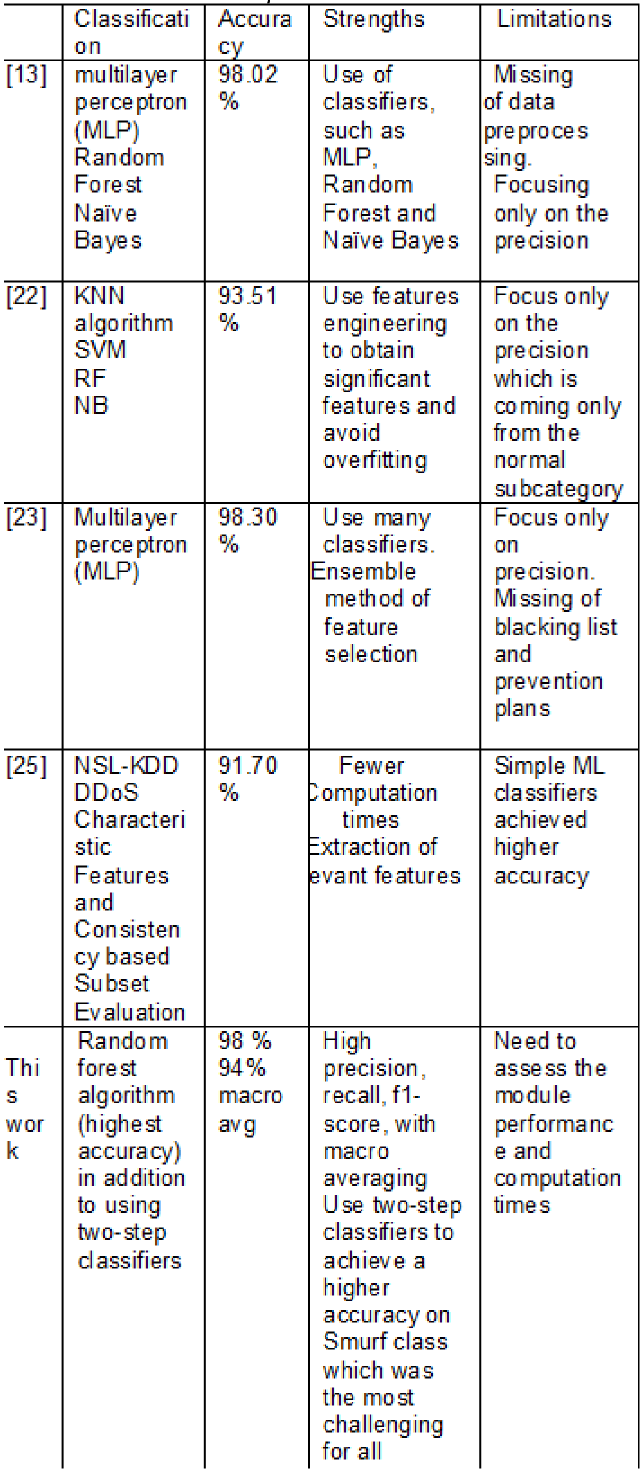
***Table 8** The Model applied to new data*

	Precision	Recall	F1-score	support
HTTP-FLOOD	0.99	0.95	0.97	226
Normal	0.99	1.00	0.99	93744
SIDDOS	0.87	0.92	0.89	302
Smurf	0.35	0.32	0.33	637
UDP-Flood	1.00	0.90	0.95	9946
Accuracy			0.98	104855
Macro avg	0.84	0.82	0.83	104855
Weighted avg	0.98	0.98	0.98	104855

Comparison with related work approaches

Comparison with related work approaches In Table [9], a comparison of this work is provided with other related works used the same datasets. However, the accuracy and precision of this work is competitive to other studies but is not sufficient, because it is coming from the only subcategory of the labels, which is mainly the normal state. Our model uses macro weighting to evaluate the metrics for the minority classes (Attacks), which is the main focus of the research. Our model proposes multiple measures to assess, such as precision, recall, f1-score, with macro averaging to ensure that the score covers the minority and the majority of classes. Also, the framework extended by integrating the attack detection with analyzing system and prevention plan using bigdata techniques. Also, it can be noted that the Smurf class was the most challenging for all classifiers due to the nature of the attack. As a result, we used two-step classifiers, which show good rates for the Smurf class.

***Table 9** Comparison with related works*



	Classification	Accuracy	Strengths	Limitations
[13]	multilayer perceptron (MLP) Random Forest Naïve Bayes	98.02 %	Use of classifiers, such as MLP, Random Forest and Naïve Bayes	Missing of data preprocessing. Focusing only on the precision
[22]	KNN algorithm SVM RF NB	93.51 %	Use features engineering to obtain significant features and avoid overfitting	Focus only on the precision which is coming only from the normal subcategory
[23]	Multilayer perceptron (MLP)	98.30 %	Use many classifiers. Ensemble method of feature selection	Focus only on precision. Missing of blacking list and prevention plans
[25]	NSL-KDD DDoS Characteristic Features and Consistency based Subset Evaluation	91.70 %	Fewer Computation times Extraction of evant features	Simple ML classifiers achieved higher accuracy
This work	Random forest algorithm (highest accuracy) in addition to using two-step classifiers	98 % 94% macro avg	High precision, recall, f1-score, with macro averaging Use two-step classifiers to achieve a higher accuracy on Smurf class which was the most challenging for all	Need to assess the module performance and computation times

CONCLUSION

In this paper, we applied real-time big data processing and machine learning to detect traffic anomalously. Big data technologies are used to detect application and network layer DDoS attacks by integration, correlation, and classification of the information generated from different server logs. Expectedly, it will reduce the time taken to analyze and process this data in real-time [26]. Machine learning had reduced the shortcomings and challenges of DDOS detection on the application layer, which comes as a normal connection to the existing approaches in detecting anomalous behaviors. Moreover, we developed a model for our approach. Furthermore, the experiment evaluates the proposed framework accuracy and efficiency in the detection rate with different Classifiers and retains the maximal prediction accuracy with high Macro average scores. Our future work will include identifying patterns for other attacks and testing the ability of the model to detect in addition to evaluating the performance and the computational time of the framework [27,28]. Also, it can be noted that the two-step classifier has a promising result. Therefore, we are looking forward to testing one versus all classifiers for each class[29]. It is expected to have a classifier for each class or a classifier that would work on the majority class and a classifier for each of the minority class, which will lead to achieving better results.

REFERENCES

[1] Alomari, Esraa & Manickam, Selvakumar & Gupta, B B & Karuppayah, Shankar & Alfaris, Rafeef. (2012). Botnet-based Distributed Denial of Service (DDoS) Attacks on Web

[2] Servers: Classification and Art. International Journal of Computer Applications. 49. 10.5120/7640-0724. https://www.kaspersky.com/about/press-releases/2019_summertime-and-the-ddos-is-easy-q2-saw-18-percent-rise-in-attacks-compared-to-last-year

[3] Mitrokotsa,A.,Douligeris,C.:DenialofServiceAttacks, Network Security:CurrentStatusandFutureDirections,pp.117–134.Wiley, Hoboken (2006)

[4] Obaid, Hadeel & Abeed, Esamaddin. (2020). DoS and DDoS Attacks at OSI Layers. 1-9. 10.5281/zenodo.3610833.

[5] Al Jallad, K., Aljnidi, M. & Desouki, M.S. Big data analysis and distributed deep learning for next-generation intrusion detection system optimization. J Big Data 6, 88 (2019).

[6] Al-kasassbeh, Mouhammd & Al-Naymat, Ghazi & Hassanat, Ahmad & Almseidin, Mohammad. (2016). Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. International Journal of Advanced Computer Science and Applications.

[7] Medeira, Paul & Grover, Jyoti & Khorajiya, Moin. (2019). Detecting Application Layer DDoS Using Big Data Technologies.

[8] Zeb, Khan & Baig, Owais & Asif, Muhammad. (2015). DDoS Attacks and Countermeasures in Cyberspace.

[9] Vanparia, Pradip & Ghodasara, Yogesh & Donga, Mr. (2015). Network Protocol Analyzer with Wireshark. developeriq.in.

[10] Landsat, S., Khoshgoftaar, T.M., Richter, A.N. et al. A survey of open source tools for machine

learning with big data in the Hadoop ecosystem. Journal of Big Data 2, 24 (2015).

[11] Marwa Hussien Mohamed and Mohamed Helmy Khafagy. "Hash semi cascade Join for joining multi-way map reduce." 2015 SAI Intelligent Systems Conference (IntelliSys)(2015): pp.355-361.

[12] Mina samir shnoda,mohamed helmy khafagy,samah ahmed senbel,"JOMR: Multi-join Optimizer Technique to Enhance Map-Reduce Job",The 9th International Conference on INFOrmatics and Systems (INFOS2014),pp:80-86

[13] Alkasassbeh, M., Al-Naymat, G., Hassan, A.B., Almseidin, M.: Detecting distributed denial of service attacks using data mining techniques. Int. J. Adv. Comput. Sci. Appl. 7(1), 436–445 (2016)

[14] PATRO, S GOPAL & Sahu, Kishore Kumar. (2015). Normalization: A Preprocessing Stage. IARJSET. 10.17148/IARJSET.2015.2305.

[15] Goutte, Cyril & Gaussier, Eric. (2005). A Probabilistic Interpretation of Precision, Recall, and F-Score, with Implication for Evaluation. Lecture Notes in Computer Science. 3408. 345-359. 10.1007/978-3-540-31865-1_25.

[16] Behal, Sunny & Saluja, Krishan. (2017). Characterization and Comparison of DDoS Attack Tools and Traffic Generators -A Review. International Journal of Network Security. 19. 383-393. 10.6633/IJNS.201703.19(3).07).

[17] Uramova, Jana & Segeč, Pavel & Moravcik, Marek & Papan, Jozef & Kontsek, Martin & Hrabovsky, Jakub. (2018). Infrastructure for Generating New IDS Dataset. 603-610. 10.1109/ICETA.2018.8572201.

[18] Marwa Hussien Mohamed, Mohamed Helmy Khafagy, Heba Elbeh, Ahmed Mohamed Abdalla (2019). "Sparsity and cold start recommendation system challenges solved by hybrid feedback", International Journal of Engineering Research and Technology, 12 (12), 2735–2742.
Marwa Hussien Mohamed, Mohamed Helmy Khafagy, Mohamed Hasan Ibrahim (2020). "Two recommendation system algorithms used SVD and association rule on implicit and explicit data sets," International Journal of Scientific and Technology Research, 9 (1), 17–24.
Cutler, Adele & Cutler, David & Stevens, John. (2011). Random Forests. 10.1007/978-1-4419-9326-7_5

[19] https://github.com/joblib/joblib-spark

[20] Aamir, Muhammad, and Mustafa Ali Zaidi. "DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation." International Journal of Information Security (2019)

[21] Marwa Hussien Mohamed, Mohamed Helmy Khafagy, Mohamed Hasan Ibrahim (2019). "Recommender Systems Challenges and Solutions Survey", Proceedings of 2019 International Conference on Innovative Trends in Computer Engineering, ITCE 2019, February, 149–155. https://doi.org/10.1109/ITCE.2019.8646645.

[22] Singh, K.J., De, T.: Efficient classification of DDoS attacks using an ensemble feature selection algorithm. J. Intell. Syst (2017)

[23] Yusof, A.R., Udzir, N.I., Selamat, A., Hamdan, H., Abdullah, M.T.: Adaptive feature selection for denial of services (DoS) attack. In: IEEE Conference on Application, Information and Network Security (AINS), IEEE, pp. 81–84 (2017)

[24] Mostafa, Mohamed & Elkadi, Hatem & Khafagy, Mohamed. (2019). A STUDY ON THE BIG DATA LOG ANALYSIS: GOALS, CHALLENGES, ISSUES, AND TOOLS. International Journal of Artificial Intelligence and Soft Computing. 7. 5-12.

[25] Kaseb, Mostafa & Khafagy, Mohamed & Ali, Ihab & Saad, Elsayed. (2018). An improved technique for increasing availability in Big Data replication. Future Generation Computer Systems. 91. 10.1016/j.future.2018.08.015, pp. 493-505

[26] Ebada Sarhan, Atif Ghalwash, and Mohamed Khafagy.(2009). "Queue weighting load-balancing technique for database replication in dynamic content web sites." In Proceedings of the 9th WSEAS international conference on Applied computer science (ACS'09). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 50–55.

[27] Marwa Hussien Mohamed, Mohamed Helmy Khafagy, Mohamed Hasan

[28] Ibrahim. "Hash Semi Join Map Reduce to Join Billion Records in a

[29] Reasonable Time." Indian Journal of Science and Technology, vol. 11, no. 18, Jan. 2018, pp. 1–9., doi:10.17485/ijst/2018/v11i18/119112.