International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-4, April 2019 E-ISSN: 2347-2693

Machine Learning DDoS Detection Using Stochastic Gradient Boosting

M Devendra Prasad[1*], Prasanta Babu V[2], C Amarnath [3]

[1,2,3] Bharat Sanchar Nigam Limited[BSNL], Bangalore, India

[*]*Corresponding Author: devendraprasad@bsnl.co.in, Tel.: +91-08022028954*

DOI: https://doi.org/10.26438/ijcse/v7i4.157166 | Available online at: www.ijcseonline.org

Accepted: 19/Apr/2019, Published: 30/Apr/2019

Abstract— DDoS (Distributed Denial of service) attacks emerge as the most devastating attacks of all time for organizations and ISPs of all sizes. The increasing availability of DDoS-for-hire services and the proliferation of billions of Unsecured IoT devices and botnets contributed to a significant increase in DDoS attacks. These attacks continue to grow in magnitude, frequency, and sophistication. The legacy methods like signature-based detection and scrubbing are challenged, as attacks are growing smarter day by day and evading IDS. The next-generation security technologies also cannot keep pace with the scale of attacks targeting organizations. Even anomaly-based detection is suffering from many limitations with accuracy and false positives by demanding human intervention. This is our attempt to obviate manual analysis in anomaly-based DDoS detection by achieving perfect accuracy with zero misclassifications. In this paper, we demonstrated DDoS anomaly detection on the open CIC datasets using Stochastic Gradient Boosting (SGB) machine learning (ML) model. Using this ML model and by meticulously tuning hyperparameters, we achieved maximum accuracy and compared the results with other machine learning algorithms.

Keywords— DDOS attacks, anomaly detection, machine learning, stochastic gradient boosting, scikit-learn, XGBOOST

I. INTRODUCTION

DoS attacks cause denial of service to legitimate requests by exhausting the resources of network and services. To maximize the impact, the attack will be launched from distributed sources, called distributed denial of service attack. In the majority of the cases, these attacks are launched by botnets. The largest DDoS attack on the latest records happened on Feb 2018 as revealed by git hub. The attack originated from over a thousand different autonomous systems across tens of thousands of unique endpoints. It was an amplification attack using the Memcached-based that peaked at 1.35Tbps. Another major DDoS attack is the Mirai[1] botnet which was used in a high volumetric DDoS attack around 1.1 Tbps that took down a large part of the internet in October 2016. Mirai successfully commanded nearly 100,000 bots by exploiting poor security of cameras, home routers, DVRs, and printers with default credentials used for their telnet ports. This was not an infection—just poor security policy and lack of attention by the vendors.

By the nature of exploitation DDoS attacks can be subsumed under 3 categories namely application-level attacks, protocol attacks, and volumetric attacks. We will cover the literature of some of them under each category which is relevant to the context of our work and dataset.

I.I Application-level attacks:

These are the low volumetric attacks that exploit the vulnerabilities in layer 7 protocols like HTTP. Application attacks are the trickiest of the DDoS attacks as they are harder to identify and mitigate. These type of attacks are the most sophisticated and stealthy attacks because they can be very effective with a single machine generating traffic at a low rate can crash the web server. This makes these attacks very difficult to proactively monitor with traditional flow-based monitoring solutions like IDS (intrusion detection system).

The most common attacks under this category are low-and-slow attacks such as Slowloris, Slow HTTP POST and Slow Read attack by exhausting concurrent connections pool, GET/POST floods, and Apache Range Header attack by raising considerable memory and CPU usage on the server.

Slowloris attack:

Slowloris and Slow HTTP POST DoS, targets the design of HTTP protocol, which expects requests to be completely received by the server before they are processed. If an HTTP request is incomplete, or packets are received slowly, the resources which are reserved for the incomplete requests are held busy waiting for the rest of the data. In this process, if all available resources are allocated to such type of requests such that it can no more handle the new requests is nothing but denial of service. This tool targets HTTP server by

sending HTTP headers to server slowly (maximum allowable time by the server) and by without completing the request.

HTTP GET/POST flood attack:
In this form of attack, distributed clients are coordinated to send multiple HTTP GET requests for different types of content like files, images or some other web resources from a targeted server. When the target is flooded with requests and responses, the legitimate requests are denied causing denial of service.

In the same way, the server can be flooded with HTTP POST requests. If POST requests involve in database write operation the impact of attack will be more as compared to HTTP GET as more intensive operations will be involved which demands a high amount of processing power and bandwidth. So by sending flood of POST requests, the capacity of the target server can be easily saturated to cause denial-of-service.

I.II Protocol attacks:
These attacks target the vulnerabilities in Layer 3 and Layer 4 implementations of systems. Includes SYN flood attacks, fragmented packet attacks, Ping of Death, Smurf attacks. This type of attack consumes server resources or network bandwidth.

SYN flood attacks:
This attack exploits the weakness in the TCP connection sequence. In a normal connection scenario, TCP connection is established by a 3-way handshake between client and server by exchanging TCP packet sequence with SYN, SYN-ACK, ACK flags starting from client end. In the case of the SYN flood, the attacker sends multiple SYN packets without responding with SYN-ACK.TCP SYN can also be sent from source spoofed IP addresses. In either way, the server continues to wait for SYN-ACK by reserving resources for each of request, starving out resources for new connection which ultimately results in denial of service.

Ping of death:
In this case, the attacker sends multiple malformed or malicious pings to target. The maximum length of IP packet 65,535 bytes poses MTU limitations on datalink layer which splits packet into fragments of 1500 bytes. This need to be reassembled at target host which can cause buffer over flow causing denial of service for benign packets.

Smurf attack:
It is executed by Smurf malware. It builds a spoofed packet by setting source IP to the IP address of the target victim. The attack is amplified by sending ping packets to IP broadcast address of an intermediate network. Each host connected to the intermediate network responds with ICMP echo reply packets to target .it makes the target potentially overwhelmed and resulting denial of service to legitimate traffic.

I.III Volumetric Attacks:
In these type of attacks, network resources and network bandwidth are starved out to deny legitimate services by flooding UDP traffic. To hide the source identity, reflection is used by spoofing source IP and amplification is used to multiply attack traffic bandwidth through services like DNS, NTP, and Memcached. The primary principle of any amplification attack is to send a spoofed request to UDP services and eliciting the larger amount of data as a response which is many folds higher than the size of the request.
Under this category, the major part of attacks covered by reflection and amplification attacks and ICMP floods.
Popular amplification and reflection attacks are:

Memcached Amplification:
One of the most recently evolved attacks that successfully reached the amplification factor of 51000. Memcached is a tool used to cache data to make data processing faster. It is only intended to be used on systems that are not connected to the internet, as there is no authentication mechanism. However, there are currently more than 50,000 known vulnerable systems, according to Akamai. The attacker spoofs request to a vulnerable UDP Memcached server, which then floods a targeted victim with a large response, potentially overwhelming the victim's resources. While the target's internet infrastructure is overloaded, new requests cannot be processed and regular traffic is unable to access the internet resource, resulting in denial-of-service.

DNS Amplification:
This attack makes use of vulnerabilities of DNS systems which are publicly accessible and supports open recursive relay. The attacker spoofs the source IP with the victim's target IP and queries DNS server. When DNS server is queried for "ANY" resource record, in a single request all zone information is returned and it is reflected to target.

The attack can be amplified even by EDNS (Extension Mechanisms for DNS) or DNSSEC (The Domain Name System Security Extension). Through these methods request message of 60 bytes can be converted to response message of around 4000 bytes to target victim, achieving 1:70 amplification factor. When this attack is launched by Botnets, where thousands of Bots query, multiple DNS servers which in turn send response data to the victim, increase the volume of traffic by many folds and accelerates the rate at which the Target server resources will be depleted.
In March 2013, The Spamhaus Project was targeted by a massive DDOS attack. It was declared that DNS Amplification was the primary tactic exploited by the attackers.
NTP Amplification:

Primarily NTP was designed for clock synchronization between the internet connected systems. In the most basic type of NTP amplification attack, an attacker repeatedly requests "get monlist" from NTP server, with the spoofed IP address. The NTP server responds with a list of the last 600 hosts that connected to the queried server. In an NTP amplification attack, the query-to-response ratio ranges between 20:1 to 200:1 or more. These attacks more prevalent as the list of open NTP servers can be easily obtained through a tool like Metasploit or data from the Open NTP Project with which attacker can easily generate a devastating high-bandwidth, high-volume DDoS attack.

I.IV Challenges in traditional detection methods:

Much of the DDoS bots remediation activity today is still a very manual process. Bots using certain IP addresses or domains are identified, then necessary steps are taken to block them at the proxy or firewall.

As the sophistication level of malicious bots and other attacks increases, traditional approaches to security become less effective.

When it comes to detection of DDoS attacks targeting web applications and network bandwidth, traditional approaches such as Poll-Based Monitoring, Flow-Based Network Parameter, Deep Packet Inspection, Frequency-Based Detection, have their own limitations as these approaches rely on the signature of the attacks.

Signature for an attack cannot be developed on its own. Human intervention is needed for each attack to be modeled. Moreover, it will take considerable time and effort to develop a signature and apply it, as a rule, to catch and stop a known attack. To defeat signature-based approaches attackers create slightly different variants to bypass IDS. This is the reason for the proliferation of DDoS botnet variants today.

I.V Detection using ML:

ML provides a nonlinear way to identify attacks, looking beyond simple signatures, identifying similarities to what has happened before, and marking things that appear to be anomalies.

ML can greatly improve detection and defense capabilities by using external threat intelligence about DDoS bot behaviors and combining it with data collected about real traffic samples to learn about new bot patterns. This information is then fed into a ML solution. After the ML solution consumes the various data points, it can be told to run multiple models whereby the human provides training input in an active feedback loop approach. ML solutions, in turn, can launch automated processes for blocking bot traffic based on the machine's new understanding of what type of bot traffic to now look for. ML is required to identify unusual behavioral patterns and bring them to the attention of analysts. More important, for common and repeated suspicious behaviors, organizations can use ML to automatically block the traffic and alert an analyst that the problem has been resolved.

AI and ML techniques are exceptionally useful when it comes to observing, quantifying, and classifying inbound requests based on the degree of maliciousness.

I.VI Motivation for our work:

Currently, security products are already available in the market that adopted supervised ML. For most of these products, manual input is provided to the ML engines that are at work scanning massive numbers of log entries to identify anomalous behaviors. The supervised ML system augments the manual input as it's trained to improve detection of "significant events" in the logs and to immediately bring those events to the attention.

Even though ML capable devices are in practice today, still, they are demanding analyst intervention, to analyze the response of detection engine before feeding into blocking engine to segregate false positives. In this paper we attempted to engineer the SGB algorithm to detect DDoS attacks without any misclassifications, making the detection system fully automatic.

Rest of the paper is organized as follows, Section II contains the related works of open dataset generation and DDoS detection methods. Section III covers our proposed methodology which focuses on dataset description, theoretical background of SGB algorithm and its implementation. Section IV describes the performance of SGB compared to other algorithms . Section V is about our future work and conclusion.

II. RELATED WORK

Machine learning, in the domain of network security, is an active area of research in both industry and in academia. We cover some of the research works which are the foundation of our work and some of the works which are in the same lines.

One of the datasets, CICIDS2017[2], from which we extracted DDoS traffic is explained in detail in [2]. They evaluated generated dataset using seven ML classifiers out of which best results are obtained by using Random Forests in terms of precision and execution time. In [3] authors proposed a novel detection approach for application layer DoS attacks based on nonparametric CUSUM algorithm and tested. It mainly concentrates on detection of application layer DoS attacks based on different types of sampling attacks. The evaluation dataset created by generating different types of application-layer DDoS attacks and mixed

with benign traffic generated by [4]. The same dataset is used as one of the three datasets to make our final dataset.

Generally, DDoS detection solutions are employed at network of victim but in [5] authors made a proposal to detect DDoS at source end in cloud environment using statistical data from both the cloud server's hypervisor and the virtual machines, to prevent network packets from being sent out to the outside network which is evaluated by nine ML algorithms and they got best prediction results with Random Forest Model. Strong defense mechanism against network security attacks are employed at server or network end but, no dedicated security controls at end-user devices on the internet. This sort of vulnerable devices attracted attackers easily and becoming part of botnets. In [6], a model is proposed to detect attacks from end-user IOT devices with high accuracy based on IoT-specific network behaviors using variety of ML algorithms including neural networks.

Although all these studies addressed the need to identify DDoS attacks in a more intelligent way and reported the shortcoming by different methods none of the works demonstrated the performance results with SGB to evaluate DDoS datasets and accuracy is not improved more than as that of Random Forest. More overall models are evaluated over datasets having relatively fewer samples compared to our dataset. We improved the performance to reach 100 percent, by using a dataset of diversified DDoS flows with more than 10 million bidirectional flows and meticulously tuning the hyperparameters.

III. METHODOLOGY

The prediction of DDoS attack using proposed algorithm is achieved through the following stages.

III. I BUILDING DATASET & FEATURE EXTRACTION:

The dataset used to train our proposed model is extracted from three open data sets published by CIC Canada [2][3]. The details are summarized in table 3.1. The reason to combine multiple data sets is to simulate the near real-time DDoS traffic by introducing diversity, as each dataset captured in different years (2016,2017,2018) using various attack tools in a different network and system setup. The actual data of datasets contain many attacks other than DoS and DDoS also. So we extracted only DoS and DDoS traffic files as mentioned in Table 3.1.

Table 3.1 Final dataset extraction details

Data Set	File name	Tools/Attack type	File Size (GB)
CSE-CIC-IDS2018-AWS	Friday-16-02-2018/(all pcaps)	DoS-SlowHTTPTest DoS-Hulk	47
	Thursday-15-02-2018/(all pcaps)	DoS-GoldenEye DoS-Slowloris	46
	Wed-21-02-2018/(all pcaps)	DDOS-LOIC-UDP DDOS-HOIC	76
	Tues-20-02-2018/(all pcaps)	DDoS attacks-LOIC-HTTP DDoS-LOIC-UDP	52
	Thuesday-20-02-2018_TrafficForML_CICFlowMeter.csv		
CICIDS2017	Monday-WorkingHours.pcap	Benign Traffic	10.8
	Friday-WorkingHours.pcap	DDos-LOIC	8.8
		Port scan	
CIC DoS dataset(2016)	AppDDos.pcap	slowbody2 ddosim	4.6
		goldeneye slow headers	
		hulk slowloris	
		rudy	
		slowread	

The nature of attack and Simulations tools used to generate different types of DDoS traffic against each file is also mentioned in table 3.1.

The nature of files we extracted from datasets are .pcap files which contain raw attack traffic captured at the network level. Though processed .csv files readily available to train ML models, those are missing some features and number features are not uniform across the datasets. So we extracted bidirectional flows into .csv files from .pcap files using open source CCFLOWmeter-v4[2] by which 84 features are extracted. The most important features out of 84 features are provided in figure 4.5 and discussed under the section "feature importance".

The dataset CSE-CIC-IDS2018-AWS[2] contain 4 days of DDOS traffic and each day traffic is generated by two distinct DDoS tools. Each day of data consists of around 500 .pcap files. Each file is converted into .CSV file and merged all files to a single file of the day and labeled it with the name of a type of attack. Type of attack traffic is identified based on Source IP of attack and victim systems mentioned in [2]. By following the same procedure all 4 days of data is labeled and merged into a single file. The same procedure is followed for other 2 datasets CICIDS2017[2], CIC DoS[3] and generated two corresponding aggregated files. Finally, the three aggregated multiclass labeled files are combined into the single file, with bidirectional flows amounted to 6.4 Million. As our ML model dealing with binary classification, all multi-class labels are relabelled to name "DDoS".

In the next step, we extracted benign data from two sources first, from "Monday-WorkingHours.pcap", extracted to .CSV and labeled as "Benign". Second, from "Tuesday-20-022018_TrafficForML_CICFlowMeter.csv". All benign flows are combined to 7.9 Million.

To construct a balanced dataset, 80% of Benign flows are sampled out of 7.9M so that final benign data points count to 6.3 M. These benign blows merged with 6.4 M DDoS flows as extracted previously to create a final balanced dataset with 12.79 M data points.

We created one imbalanced dataset to simulate real-time attack volume proportion. In the case of volumetric DDoS attacks, attack traffic is more than benign traffic but in the case of application-level DDoS attacks, such as slowlowris attack traffic is in smaller proportions compared to benign traffic. As the majority of data points in data set are related to application level DDoS traffic we created an imbalanced dataset by sampling 20% DDoS traffic and combining with 80%of benign traffic. Total count of flows in each dataset is shown in table 3.2

Table 3.2 Flow details in balanced & imbalanced datasets

Dataset	Label:DDoS	Label:benign	Total Flows
Balanced	6472647	6321980	12794627
Imbalanced	1294529	6321980	7616509

III.II The ML classifier for detection:
We used SGB[7] algorithm to detect DDoS attacks which are explained and implemented as follows.

III.IIIBackground of Stochastic Gradient boosting algorithm:
Gradient boosting[8] is one of the most powerful algorithms for building predictive models. It is an ensemble learning method, which combines the predictive power of individual weak learners to boost the accuracy of the final model.

An ensemble is a group of base models which can be of any ML model. But in real-time applications, where latency is an important factor, Decision trees are used as base models of the ensemble. Bagging is another type of ensemble model that uses Decision Trees. In Bagging base learners are trained with random subsamples of actual dataset and prediction of majority learners is considered as the final label. As the decision of each individual model directly contributes to the final result, base learners can't be weak learners. In other words, each tree needs to grow maximum depth to result in accurate prediction. So trees in ensembles of bagging will have higher variance.

In contrast, trees in boosting ensemble have low variance and high bias. Depth of the tree is smaller, even shallow trees can serve the base learners. So the base learners in boosting are weak learners and predictive power is very weak. Each of the weak learners will not contribute directly to the final decision, instead, they act on error or residual of the previous model and train the overall learner with low bias and low variance.

Boosting operation is performed through the series following steps:
1. For the given dataset first, the initial model(F_0) naively predicts the label γ which results in error or residual F_0 -y

2. New model h_1, instead of predicting the label of actual data points, it tries to fit residual in the step1.
At this step overall model can be formulated as

$$F_1 = F_0 + h_1$$

where F_1 is a boosted version of F0, means better prediction.
3. Now error in the previous step is F_1-y, and h_2 is the function which tries to fit residual.
Now, F_1 & h_1 can be combined to give function F_2, a better v ersion of F_1

$$F_2= F_1 + h_2$$

This process continues for n iteration until error is minimized and the desired accuracy is achieved.
After n iterations,

$$F_n= F_{n-1} + h_n$$

The additive learner introduced at each stage try to reduce the error from the previous stage and do not disturb previous learners. In boosting operation "h_n" tries to fit residual or Loss function, but in gradient boosting it fits gradient loss of function.

Gradient boosting algorithm [8] over training data set $\{x_i, y_i\}_{i=0}^{n}$ over "m" iterations take the following steps.

The initial approximate loss function is,

$$\text{Loss function}=L(y_i, F(x_i))$$

1. When the model is started with some initial naïve prediction γ, a naïve initial prediction of $F(x_i)$,

$$F_0(x)= \text{argmin}_\gamma \sum_{i=0}^{n} L(yi, \gamma)$$

2. The pseudo residual R, of i^{th} data point at m^{th} iteration is the gradient of loss function at i,m

$$R_{im} = -\alpha[\partial L(y_i, F(x_i))/\partial(F(x_i)]_{F(x)= Fm-1(x)}$$

where α is a learning rate.

3. Fit $h_m(x_i)$ to above-derived residuals i.e train additive model over the data set $\{x_i, R_{im}\}_{i=0}^{n}$

4. Compute γ_m by solving the following equation

$$\gamma_m = \text{argmin}_\gamma \sum_{i=0}^{n} L(yi, F_{m-1(xi)}) + \gamma h_m(x_i)$$

5. At the end of each iteration add it to the final model

$$F_{m(x)} = F_{m-1(xi)} + \gamma_m h_m(x_i)$$

For each node, there is a factor γ_m with which hm(x) is multiplied. This accounts for the difference in the impact of each branch of the split. Gradient boosting helps in predicting the optimal gradient for the additive model, unlike classical gradient descent techniques which reduce error in the output at each iteration.

In the above algorithm (Gradient Boosting) for each additive base learner is trained with the residual dataset over full data points but in the case of stochastic gradient boosting algorithm, at each iteration a subsample of the training data is drawn at random (without replacement) from the full training dataset. The randomly selected subsample is then used, instead of the full sample, to fit the base learner.

III.IV Implementation:
Stochastic Gradient Boosting algorithm is implemented by XGBOOST, an open-source software library which provides a gradient boosting framework in python.

XGBOOST simulates gradient boosting algorithm with default parameters. To achieve stochastic gradient boosting we need to specify what fraction of data set need to sample randomly for each additive base learner to be trained at each iteration. As random sampling can be done row-wise, column-wise and split-wise (when building tree), we need to set values for XGBOOST parameters, namely "subsample", "colsample_bytree" and "colsample_bylevel" respectively. Along with these parameters, the maximum number of trees in the ensemble, depth of each tree and learning rate of gradient also need to be set with optimal values. These parameters can be set by specifying values for "n_estimators", "max_depth" and "learning_rate" respectively. For tuning these hyperparameters, we used Grid search technique with 3-fold cross-validation. The final Model is implemented with the tuned hyper-parameters returned by Grid search.

The dataset is divided in to train and test datasets in 2:1 ratio and model is trained on train dataset and tested with test dataset. SGB model has trained with imbalanced dataset also in the same way.

IV. RESULTS AND DISCUSSION

The performance of SGB is compared with the following ML models

1. K Nearest Neighbours using "kd_tree" algorithm (K-NN)
2. Naïve Bayes with Bernoulli event model(NB)
3. Decision tree using Gini impurity (DT)
4. Random Forest using Gini impurity (RF)

The above algorithms are implemented using Scikit-learn python library.

All ML models are trained over both balanced dataset and imbalanced dataset. The following classification metrics are used to evaluate the proposed model and compare with the above ML algorithms.

Accuracy:

It is the ratio of correctly classified data points to the total number of data points in a given data set.
(TP + TN)/ (Total data points).

Precession:

It is the ratio of all correctly classified items to all actually classified items.
Defined as, TP/ (TP + FP).

Confusion Matrix:

In binary classification, it is the 2x2 matrix. In our case it is described as below:

Label	Predicted as "ddos"	Predicted as "benign"
Data points labelled with "ddos"	True Positves(TP)	False Negatives(FN)
Data points labelled with "benign"	False Positive(FP)	True Nehatives(TN)

Precession:
It is the ratio of all correctly classified items to all actually classified items.
Defined as, TP/ (TP + FP).

F1-score:
Mathematically it is defined as
$2 * TP/ (2 * TP + FN + FP)$. It computes the harmonic mean of the precision and the recall.

Recall:
It is the ratio, TP/ (TP + FN). Also called sensitivity or true positive rate.

The obtained results for all models are shown in table 4.1 along with tuned hyperparameter values and run time for each model. All models are executed in same standalone Linux Machine of 216 GB RAM and 30-Cores Intel-Xeon 2.7GHz CPU.

From the performance metrics of all models over balanced dataset from Table 4.1, It is evident that SGB outperformed all other models with zero false positives and false negatives and 100% accuracy and literally zero misclassifications. NB recorded the lowest accuracy and also other metrics. But it is the fastest model with run time, 20 secs with a single thread. K-NN is the slowest model with 5hrs run time with 25 parallel threads. When executed with a single thread, it ran for 20 hrs. DT model is the second best when the trade-off between run time and accuracy is considered as it completed its execution in 4min. Though Random Forest has less run time with 2min, it can't be compared directly as RF execution is parallelized with 30 jobs.

Table 4.1 Results over balanced dataset

Metric	SGB	RF	DT	K-NN	NB
Hyper Parameters	max_depth=5, learning_rate=0.2 ,subsample=0.4, colsample_bytree=1.0, colsample_bylevel=0.1, n_estimators=200 n_jobs=30	max_depth=5, n_jobs=30 n_estimators =200	max_depth=5,	k=6, n_jobs=25	Bernoulli
Accuracy (%)	100	99.95	99.94	99.94	91.81
F1-score(%)	100	99.95	99.94	99.95	91.73
Precision(%)	100	99.96	99.92	99.95	91.37
Recall(%)	100	99.95	99.96	99.95	89.93
Confusion Matrix	[2086461 0] [0 2135766]	[2085725 736] [978 2134788]	[2084882 1579] [756 2135010]	[2085311 1150] [1145 2134621]	[1958145 128316] [217195 1918571]
Total Mis-classifications	0	1714	2335	2295	345511
Execution Time	10 min	2 min	4 min	5 hrs	20 sec

The run time of SGB is 10 min when executed with 30 threads. It is not as fast as Random Forest where bagging operation can be computed parallel. But in real time this can be greatly reduced by deploying SGB in distributed processing environment with an n-node cluster.

The real-time attack traffic will be having high variance and results may vary for the model which is trained over the balanced dataset. To test the model against this scenario we trained SGB over the unbalanced dataset, created in section 3.1. All other models are evaluated over this dataset to compare results. SGB recorded 100 percent accuracy and zero misclassifications even with the imbalanced dataset. This proves that SGB can be engineered to be insensitive to the variance of the input data which is an essential requirement when handling real-time attack traffic. Other ML model resulted in lower performance metrics as that of balanced data set. The performance of DT, K-NN, RF are only slightly degraded but Precision of Naive Bayes exhibited huge difference from 91% to 69%. The values of all metrics are tabulated in 4.2

Table 4.2 Results over imbalanced dataset

Metric	SGB	RF	DT	K-NN	NB
Hyper Parameters	max_depth=5, learning_rate=0.2 , subsample=0.4, colsample_bytree=1.0, colsample_bylevel=0.1, n_estimators=200 n_jobs=30	max_depth=5, n_jobs=30 n_estimators =200	max_depth=5,	k=6, n_jobs=25	Bernoulli
Accuracy (%)	100	99.89	99.93	99.94	92.07
F1-score(%)	100	99.69	99.79	99.82	80.45
Precision(%)	100	99.99	99.66	99.85	69.25
Recall(%)	100	99.38	99.92	99.79	95.96
Confusion Matrix	[2086364 0] [0 427084]	[2086360 4] [2626 424458]	[2084946 1418] [334 426750]	[2085745 619] [859 426225]	[1904428 181936] [17213 409871]
Total Mis-classifications	0	2630	1752	1478	199149
Execution Time	5 min	50 sec	43 sec	1.2 hrs	17 sec

The Receiver Operating characteristic curve is another performance metric which illustrates the diagnostic ability of binary classifier. ROC curve for SGB and Naïve Bayes are shown below in Fig 4.1 & 4.2 respectively.

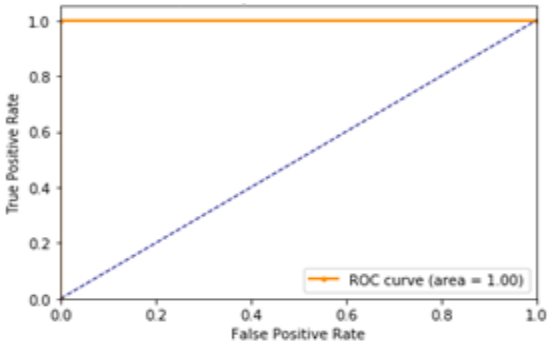

Fig. 4.1 ROC of SGB (XGBOOST) Balanced Dataset

The ROC curve for DT, RF, and K-NN is same as above as we got the area under the curve equal to one.

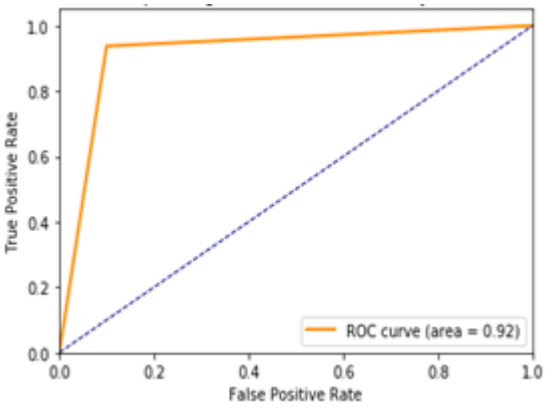

Fig.4.2 ROC of Naïve Bayes-Bernoulli (Balanced dataset)

It is found that SGB model performance unchanged even when trained with the imbalanced dataset. The ROC curve for the same is shown in Fig 4.3.

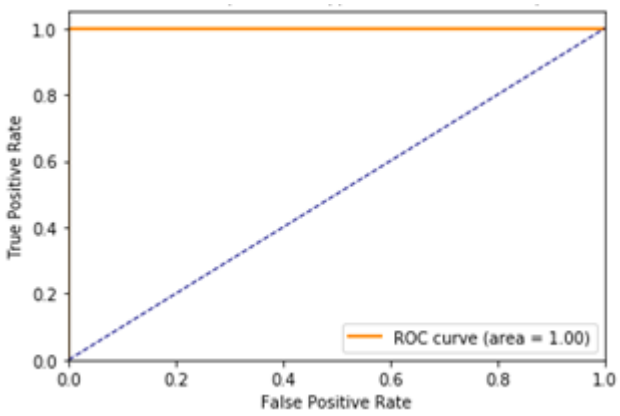

Fig 4.3 ROC of SGB (XGBOOST) Imbalanced dataset

But, the ROC curve is changed for Naïve Bayes when trained over the unbalanced dataset. The plot is shown in figure 4.4

The ROC is almost the same for RF, DT, K-NN for both unbalanced & balanced datasets. The area under the curve metric is the same. So the difference between ROC cannot be differentiated between those algorithms and look like same as in fig. 4.1 & fig 4.2. When total miss classifications are considered NB resulted in the highest number. Surprisingly for same tree hyperparameters, DT recorded better metrics compared to RF.

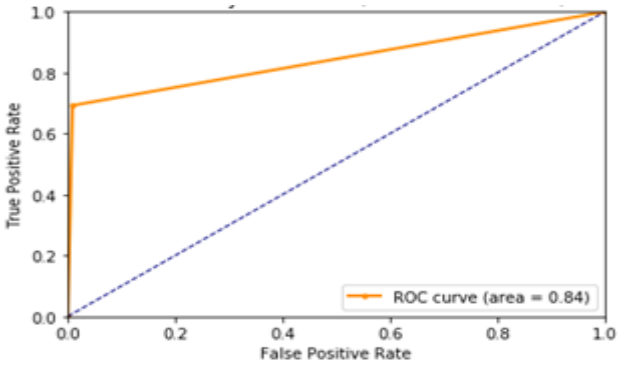

Fig 4.4 ROC of Naive Bayes (Imbalanced dataset)

4.1 Feature importance:

Advantage of using SGB is that it can automatically provide estimates of feature importance from a trained model.

Importance provides a score of the feature. Higher the score, major the role in making a decision in building a tree. The top 10 important features returned by trained SGB model are shown in Fig 4.5

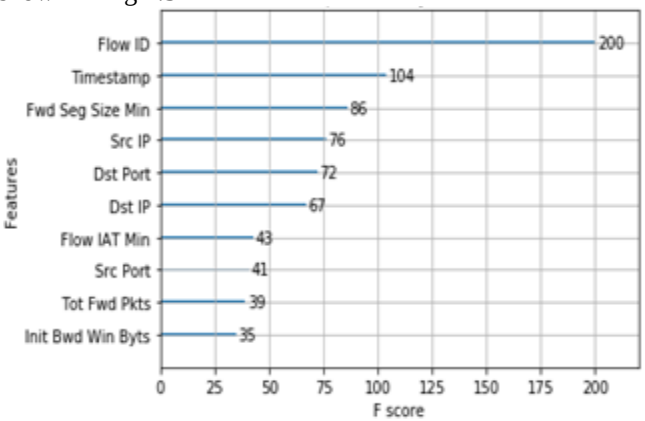

Fig 4.5 Feature importance by SGB (XGBOOST)

and also the top features given by Random Forest are shown in Table 4.3. After comparing the features returned by both models, features given by SGB are found more relevant. These are the features which played a key role in making the decision and contributing to 100% accuracy. Flow ID is the top feature in both models. This is obvious because "Flow ID" in the dataset is the combination of Source IP, Destination IP, Source Port and Destination Port and protocol (TCP or UDP) which describes the entire flow.

Table 4.3 RF Feature importance (Balanced dataset)

Label	Score
Flow ID	0.226904
Src IP	0.105322
Fwd Seg Size Min	0.094147
Dst IP	0.090014
Timestamp	0.056993
Init Bwd Win Byts	0.050132
Init Fwd Win Byts	0.034673
Bwd Header Len	0.019799
Fwd Act Data Pkts	0.01799
Fwd Seg Size Avg	0.017863

Even in real time manual analysis this parameter influences decision. "Timestamp" is the second highest feature in SGB model which is not important in real-time DDoS analysis. But in the dataset in which model is trained, the attack is generated on particular days and benign traffic in another day. So here the key role of "Time stamp" is justified.

When the model is trained on the imbalanced dataset we find that important features are changed but 100% accuracy retained. These features are presented on Fig4.6 and table 4.4 It is observed the feature minimum flow inter-arrival time, "Flow IAT Min" which has considerable importance in DDoS detection is not presented by Random Forest model but returned SGB model for both the datasets balanced and imbalanced.

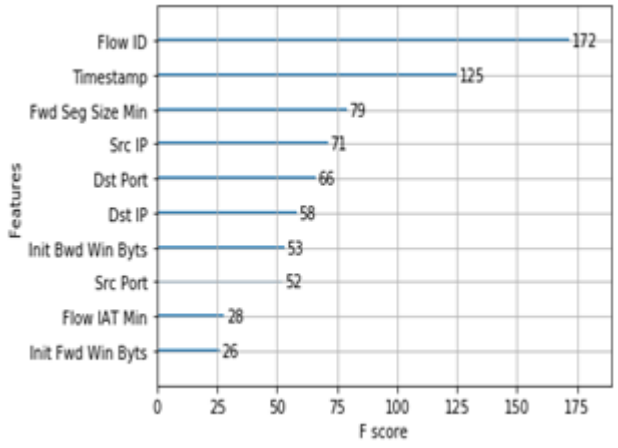

Fig 4.6 Feature importance by SGB(XGBOOST)imbalanced dataset

Table 4.4 RF feature importance (Imbalanced dataset)

Label	Score
Flow ID	0.212814
Fwd Seg Size Min	0.089639
Timestamp	0.080945
Src IP	0.06383
Fwd Pkt Len Mean	0.061426
Dst IP	0.057134
Fwd Pkt Len Max	0.039168
Fwd Seg Size Avg	0.035354
Init Bwd Win Byts	0.033372
Init Fwd Win Byts	0.024952

V. CONCLUSION AND FUTURE SCOPE

Conclusion:

In this paper, we proposed a methodology to detect DDoS attacks by using Stochastic gradient boosting algorithm.

We trained our proposed model on the hybrid dataset extracted from three open datasets and compared the results with other popular machine algorithms. It is shown that SGB surpassed K-NN, Decision trees, Random forest, and Naive Bayes by achieving 100% performance metrics. Even in case of feature importance, SGB presented the more relevant features compared to Random Forest.

The proposed model is also trained over imbalanced dataset to simulate real-time traffic and results are demonstrated. Even in this case, SGB achieved zero misclassifications and perfect evaluation metrics. Whereas metrics of other models recorded lower compared to as that of balanced dataset case.

Finally, in the context of our work, we have shown that by adapting SGB, DDoS detection system can be fully automated obviating human intervention.

Future work:

This publication is the prototype of ongoing work in our organization to detect DDoS attacks originating from customer end devices like home routers which are not protected. The real-time flows are extracted from net flow data collected at the internet gateways. The processed net flow data is labeled based on threat intelligence inferred from the data points of security controls deployed at various levels such as firewalls, intrusion protection systems, intrusion detection systems, sandboxes, web application firewalls, and privileged access management solutions.

To scale the model to large real-time time data and to achieve better latency XGBOOST is deployed with spark, a distributed in-memory, cluster computing framework in a multi-node cluster setup. We are eager to check how metrics are going to be affected.

With the continuously evolving nature of attacks, an attempt to find zero day attack poses a limitation on labeled data based on internally collected threat intelligence.so we would

like to experiment with more promising machine learning techniques beyond those discussed in this paper such as deep learning.

REFERENCES

[1].M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in Proc. of USENIX Security Symposium, 2017.

[2].Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

[3].Hossein Hadian Jazi, Hugo Gonzalez, Natalia Stakhanova, and Ali A. Ghorbani. "Detecting HTTP-based Application Layer DoS attacks on Web Servers in the presence of sampling." Computer Networks, 2017

[4]. A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput.

Security 31 (3) (2012) 357–374.

[5].Z. He, T. Zhang, and R. B. Lee, "Machine Learning Based DDoS Attack Detection from Source Side in Cloud," in Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), pp. 114–120, New York, NY, USA, June 2017

[6].R. Doshi, N. Apthorpe and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, 2018, pp. 29-35.

[7].Jerome H. Friedman, (2002), Stochastic gradient boosting, Computational Statistics & Data Analysis, 38, (4), 367-378

[8].Friedman, Jerome H. Greedy function approximation: A gradient boosting machine. Ann. Statist. 29 (2001), no. 5, 1189--1232.

Author's Profile

M Devendra Prasad pursued B.tech(Electronics and communication Engineering from JNTU Hyderabad. He has been working in Datacentre of Broad Band Networks, BSNL, as a Junior Telecomm Officer for past 10 years in the domains of Networking and Network security. He attained rigorous industry experience over Network programming and network security (Firewalls, NIDS, HIDS, AAAs). Implemented DNS, DNS64, DNSV6, DHCP, DHCPv6 services on pan India scale to serve various broadband and VOIP services. Implemented Google Global Caching in BSNL ISP over 20+ locations across India. Implemented many prototypes including IPv6 broadband (end to end), NAT64 and Automatic Bot healing system. His current area of research focuses on Machine Learning & Deep learning in Cyber Security Domain (IOT botnets). Other areas of interest include Multivariate calculus, Mathematical Optimization, Algorithms, SDN, NFV, and IOT Security.

Prasanta Babu V pursed B.Tech(Electronics and Communication Engineering) from Gitam University, Visakhapatnam in 2001 and MBA from Bangalore University in 2012. He has been working in Datacenter of BBNW, BSNL for the past 13 years. He has strong industry experience in managing different network services(CGNAT, WiMAX Broadband, DSL broadband, Google peering & caching, OSS/BSS) on pan India scale, and managing Network security of Project-3 datacenter. His areas of interests include Business Analytics, VPNs and cloud computing.

C Amaranth Pursed Bachelor of Technology in Electronics and Communication Engineering from Jawaharlal Nehru Technological University, Anantapur in 2009. He is currently working as Junior Telecom Officer under Department of Telecommunications, in Bharat Sanchar Nigam Limited(BSNL), Bangalore. He has been working in Datacentre of BSNL, BBNW for the past 5 years. He has Strong Industry Experience in Administering Different Network services like CGNAT, Peering and Caching of Various Content Providers and in network security (Firewalls & NIDS). His Areas of Interest and research include Machine learning, Cloud Security, Distributed processing & Big data analytics.

