Network Intrusion Detection in Big Dataset Using Spark and Weka

Priyanka Dahiya
[1, 2] Dept of EEE
[1, 2] Anna University Regional Campus Coimbatore

Abstract- *Due to rapid growth in network applications, Network attacks and intrusions have also been increased. Nowadays Intrusion detection systems require efficient and improved detection mechanism which could detect intrusive activities and serious threat to network security. Nowadays, huge amount of data is flowing every second; hence intrusion detection task became tedious. In our research work, we have compared and evaluatedthe performance of supervised data mining techniques on UNSW-NB15network dataset and Netflow records for various attacks using two different huge sizes of datasets on Weka and standalone cluster separately using Spark.*

Keywords- Network Intrusion Detection System(NIDS); Apache Spark; UNSW –NB15 dataset, Receiver Operating Characteristi (RoC).

I. INTRODUCTION

Day by day we are becoming network and computer technology dependent. It raises the need of secure networks. We have to improve computer network security for data integrity, confidentiality and availability. But these cannot stop intrusion detection. Vulnerable computer systems and networks are required to secure just to prevent risk of unauthorized access and data theft.

Intrusion Detection System is a system which inspects all inbound and outbound network activity and it identifies suspicious or malicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system. An Intrusion Detection System scans all packets on the network and attempts to classify the traffic as intrusive or non-intrusive. Intrusion detection is the process which begins where the firewall ends [1-2]. We have discussed Literature Survey in section 2, Research Methodology is discussed in section 3and Result is discussed in section 4.

II. LITERATURE SURVEY

Samuel Marchal et al. (2014) [3], they proposed a solution to cope large data to analyze for security monitoring perspectives. They introduced a security monitoring architecture of local enterprise networks for intrusion detection and prevention and forensic analysis. They mined DNS data, NetFlow records and honey-pot data and correlated in a distributed system as big data solution. They proposed Data correlation schemes and evaluated their performance in Hadoop and Spark. They introduced a new intrusion detection architecture that was able to correlate few data sources like HTTP, DNS, IP flow etc.

They introduced a new scalable NIDSarchitecture which collects and stores honeypot data, DNS data, HTTP traffic and IP-flow records. Five big data frameworks were evaluated for security monitoring.After their performance analysis, they found that Spark and Shark were the best performers in all scenarios and so they were suitable to implement the solution.

Sung-Hwan Ahn et al. (2014) [4], author believed that intelligent new threats are increasing and previously unknown attacks cannot be detected using existing pattern matching methods like signature, rule, and black list based solutions. They anticipated bigdata analysis solution which is a solution for detecting these kinds of unknown attacks. The author proposes a bigdata system model for big data analysis technology for detection of previously unknown attacks.

In future works, author suggests the researches which must be done to classify the data by context of intrusion detection. This can lead to implement the data relation analysis methodology and abnormal behavior detection strategy. Author is hopeful for q*uantitative* and *qualitative* assessment of proposed model and performance evaluation in future.

Kleber M.M. Vieira et al. (2014) [5], they proposed IRAS, an Intrusion Response Autonomic System, using Big Data techniques for data analytics for decision taking. Also, proposed a model for autonomic intrusion detection system based on the autonomic loop, known as MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge Base).Big Data infrastructure Hadoop was used to organize the large volume of data and extract information using the Map- Reduce framework. So it could provide intrusion detection, response and self-healing in cloud environment. This paper suggested autonomic computing to provide response to attacks on cloud.

So it provides self-awareness, self-configuration and self-healing in the cloud.

Sachin Kumar et al. (2015) [6], proposed a framework for analyzing accident patterns for different types of accidents on the road. They used K modes clustering and association rule mining algorithm for this analysis. Total 11,574 accidents were analyzed which occurred on Dehradun(uttrakhand, india) during 2009 to 2014. Six clusters (C1–C6) were taken in consideration taking K=6 which were based on attributes accident type, road type, lightning on road and road feature. Association rule mining was applied on all 6 clusters to generate rules. Trend analysis results also support their methodology that performing clustering prior to analysis helps in identifying useful results.

Michael A Hayes (2015) [7], was evaluated for two real-world sensor datasets provided by a Canadian company Brampton. The framework was also evaluated against the open-source Dodgers dataset and R statistical toolbox.The proposed work identifies a contextual anomaly detection framework. It detects content and context both. The content detector determines anomalies in real-time, identifying false positives. Only tall datasets were used so in future, there is possibility to use wide datasets with large number of features and smaller number of records.

Junlong Xiang et al. (2014) [8], they have used Extreme Learning Machine algorithm which achieve a relatively high Overall Accuracy and can decrease the time of the training phase. For large data or big they proposed a massively parallel algorithm for ELM, that is MR ELM and is a MapReduce variant of ELM.Their experiment results shows that MR ELM have a good speedup and size up performance.

Juliette Dromard et al. (2015) [9], they proposed to take advantage of Hardoop and spark in order to speed up an Unsupervised Network Anomaly Detector Algorithm, UNADA. The experiments proved that execution time can be improved 13 times allowing UNADA for large datasets processing.This paper is good step for detecting network anomalies in real time on large non sampled traffic.

S. Veetil et al. (2013) [10], they have presented their intrusion detection system that runs a Naive Bayes algorithm in a distributed manner on Hadoop. The classifier in their experiment uses the Apache Hadoop and HStreaming APIs to detect intrusions in real time scenario. According to the results, training job on the homogeneous cluster was found 37% faster than the standalone Naive Bayes algorithm and the improved algorithms.

Xun-Yi Ren et al. (2013) [11], presented an Intrusion Detection System model with feature multi-classification fusion based on hadoop. In this implementation, two algorithms i.e. K-means clustering and 1V1-SVM multi-classification method are combined. They have used Map to form a key-value pair forming new classification according to the classification center. Then they have removed the duplicate values reforming a new detection model. They have used KDD CUP99 datasets and their results of testing huge dataset show that the fused classifier has more accuracy than mere classifier.

Yafei Wu et al. (2015) [12], the original anomaly detection algorithm HOTSAX works in sequential manner in standalone machines with limited computing capabilities and storage. In this paper they have mitigated this problem by proposing distributed anomaly detection algorithm using apache spark computing platform and hadoop HDFS storage. By this approach, they have mitigated the low memory problems of their algorithm.

Aris-KyriakosKoliopoulos et al. (2015) [13], they have used DistributedWekaSpark which is scalable Big Data Mining Toolkit extends basic weka and possess the power of distributed systems. Standard Weka can be used for small datasets, not large datasets because of its memory constraints (1GB) so for execution on large datasets running apache spark out of the box, and they developed DistributedWekaSpark. It is built on the top of Apache Spark which provides fast in-memory distributed and parallel processing.Their evaluation results shows that distributed weka spark is 4 times faster than Hadoop and achieves near-linear scalability on scaling workloads. They have used classification part only using 4 types of algorithms like FP Growth, Linear regression, SVM on spark and SVM on Hadoop. They evaluated on strong and weak scaling workloads on 5GB, 20GB and 80GB datasets and 8, 32 and 128 cores systems. They found that Strong scaling efficiencies on Spark approach linearity when datasets are large.

MohiuddinSolaimani et al. (2014) [14], they performed experiments on a real-time, Chi-square test based anomaly detection framework using Bigdata tool Apache Spark. They have seen that it performs Chi-square based comparisons for anomaly detection by segmenting the performance data streams of VMware virtual machines into windows of variable lengths. The performance data considers both CPU and memory usage and is scalable to data from heterogeneous sources. By the experiments, it is evident that the proposed technique is able to detecting abnormal changes in CPU utilizations of VMware virtual machines.

Tamer F. Ghanem et al. (2014) [15], they proposed a hybrid approach for anomaly detection in large datasets using genetic algorithms and multi-start meta-heuristic method. They have evaluated this approach on NSL-KDD dataset, a modified version of the KDD CUP 99 dataset. The results show its effectiveness accuracy of 96.1% which is better than other machine learning algorithms.

It was found that limited research was done on anomaly detection using Bigdata tools like apache spark on large intrusion datasets.

III. RESEARCH METHODOLY

The dataset UNSW-NB 15 is collected to experiment was done on Wekas tool and Hadoop tool.

A. DESCRIPTION OF UNSW-NB 15 DATASET

For the evaluation of performance and effectiveness of NIDS, we require a comprehensive dataset which contains both normal and abnormal behaviors. Lot of research has been done using older benchmark data sets like KDDCUP 99 and NSLKDD but these data sets do not offer realistic output performance. The reason is that KDDCUP 99 loads of redundant and missing records in the training set. So these datasets are not comprehensive representation of modern low foot print attack environment.

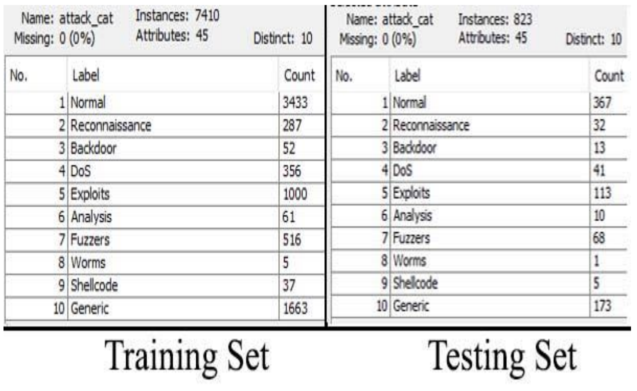
UNSW-NB 15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). It contains both real modern normal activities and synthetic contemporary attack behaviors [16].

UNSW-NB15 dataset is available in comma-separated values(CSV) file format. There are 175,341 records in training set and 82,332 records in testing set with all different 9 types attack and normal records. There are 49 attributes or features with 10 class values in this dataset. All records are divided in two major categories of the records - normal and attack. The attack category is again subdivided into 9 categories of attack types. Attack types are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.

Category	Training Set	Testing Set
Normal	56000	37000
Generic	40000	18871
Exploits	33393	11132
Fuzzers	18184	6062
DoS	12264	4089
Reconnaissance	10491	3496
Analysis	2000	677
Backdoor	1746	583
ShellCode	1133	378
Worms	130	44
Total Instances	1,75,341	82,332

Table 1: Distribution of records in UNSW NB-15 dataset

We have performed experiments on 2 sets of UNSW datasets to evaluate the performance of all classifiers.These classifiers have been evaluated first on Basic Weka tool then we evaluated these classifiers using Apache spark. Multiclass Classifier and Randomizable Filtered Classifier were used with Random Tree classifier Number of instances in dataset-1 is 7410 in training and 823 in testing. In dataset-2, instances for training are 47342 and 5260 instances are for testing.Big dataset is of approx. 602 MB. The training data is 397.32MB and test data is 198.66 MB used in Spark. Small dataset is 100 MB . Training data is 66 MB and test data is 33 MB used Weka.



Name: attack_cat Missing: 0 (0%) Instances: 7410 Attributes: 45 Distinct: 10

No.	Label	Count
1	Normal	3433
2	Reconnaissance	287
3	Backdoor	52
4	DoS	356
5	Exploits	1000
6	Analysis	61
7	Fuzzers	516
8	Worms	5
9	Shellcode	37
10	Generic	1663

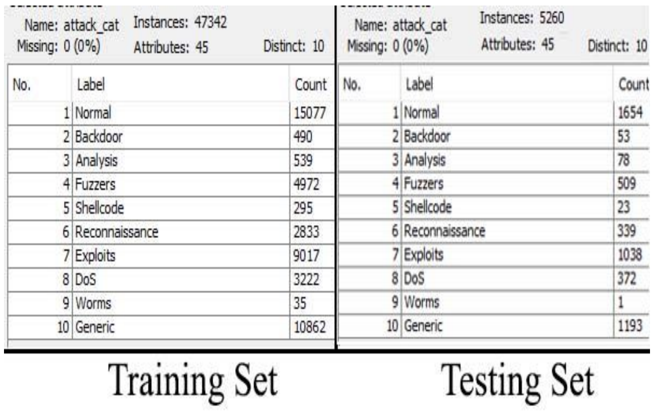
Training Set

Name: attack_cat Missing: 0 (0%) Instances: 823 Attributes: 45 Distinct: 10

No.	Label	Count
1	Normal	367
2	Reconnaissance	32
3	Backdoor	13
4	DoS	41
5	Exploits	113
6	Analysis	10
7	Fuzzers	68
8	Worms	1
9	Shellcode	5
10	Generic	173

Testing Set

Figure 1: Distribution of normal and attack instances in Dataset-1



Name: attack_cat Instances: 47342
Missing: 0 (0%) Attributes: 45 Distinct: 10

No.	Label	Count
1	Normal	15077
2	Backdoor	490
3	Analysis	539
4	Fuzzers	4972
5	Shellcode	295
6	Reconnaissance	2833
7	Exploits	9017
8	DoS	3222
9	Worms	35
10	Generic	10862

Training Set

Name: attack_cat Instances: 5260
Missing: 0 (0%) Attributes: 45 Distinct: 10

No.	Label	Count
1	Normal	1654
2	Backdoor	53
3	Analysis	78
4	Fuzzers	509
5	Shellcode	23
6	Reconnaissance	339
7	Exploits	1038
8	DoS	372
9	Worms	1
10	Generic	1193

Testing Set

Figure 2: Distribution of normal and attack instances in Dataset-2

ALGORITHM

Byes Algorithm: Bayes' rule says that if you have a hypothesis *H* and evidence *E* that bears on that hypothesis, then

$$P(\text{H} \,|\, \text{X}) = \frac{P(X|H)P(H)}{P(X)}$$

$P(H|X)$ is the posterior probability, or a posteriori probability, of H conditionedon X. In contrast, $P(H)$ is the prior probability, or a priori probability, of H. The posterior probability, $P(H|X)$ is based on more information than the prior probability, $P(H)$, which is independent of X. Similarly, $P(X|H)$ is the posterior probability of X conditioned on H. $P(X)$ is the prior probability of X. "How are these probabilities estimated?" $P(H)$, $P(X|H)$, and $P(X)$ may be estimated from the given data.

Implementation steps are as follows:

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities of generic is like = 0.29 and probability of intrusion is 0.64.

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Random Forest Algorithm: Random Forests grows many classification trees. Each tree is grown as follows:

Step 1: If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.

Step 2: If there are M input variables, a number $m \ll M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constantduring the forestgrowing.

Step 3: Each tree is grown to the largest extent possible. There is no pruning.

It is noted that algorithm of other classifiers is in build inside spark and weka tools .Here only two algorithms are discussed.

IV. RESULT

The results of smaller dataset-1 and larger dataset-2 are illustrated. Big dataset is of approx. 602 MB. The training data is 397.32MB and test data is 198.66 MB used in Spark. Small dataset is 100 MB. Training data is 66 MB and test data is 33 MB used Weka.

Result are compared using weka then they are compared using apache spark. After that we compare comparedthe results of weka and apache spark using dataset-1 and dataset-2 separately.

The performance of 10 different classifiers is evaluated on the basis of various parameters like accuracy, FPR, Training Time, precision, recall and ROC area.

Classifier	Accuracy	Kappa	Mean Absolute Error	FPR(False Positive Rate)	Precision	Recall	ROC Area	Training Time (Sec)
Naïve Bayes	54.55	0.4478	0.0903	0.024	0.766	0.546	0.881	0.23
REP Tree	89.47	0.866	0.0281	0.015	0.9	0.895	0.988	1.73
Random Tree	89.77	0.8555	0.0303	0.011	0.894	0.898	0.99	0.77
Random Forest	90.2	0.8754	0.0307	0.016	0.903	0.902	0.991	12.91
J48	88.67	0.8559	0.0325	0.018	0.889	0.887	0.984	1.7
Random Committee	87.93	0.8462	0.0319	0.021	0.881	0.879	0.986	0.27
Bagging	88.55	0.8559	0.0325	0.018	0.889	0.887	0.984	1.55
MultiClass	86.46	0.827	0.0335	0.025	0.861	0.865	0.974	1.34
Randomizable Filtered	88.15	0.8331	0.0373	0.02	0.888	0.881	0.986	1.02
IBK	89.66	0.8542	0.026	0.01	0.895	0.897	0.99	4.58

Table 2: Classifier Evaluation results from Weka (Dataset1)

It is clear from table 2 that when we used dataset-1 using weka then the Accuracy of REP Tree (89.47%), Random Tree (89.77%), Random Forest (90.2%) and IBK(89.66%) algorithms was almost same. But Random Forest (12.91 Sec) and IBK (4.58 Sec) took more training time so REP Tree (1.73 Sec) and Random Tree (0.77 Sec) were the winners in Accuracy and Training Time.

Classifier	Accuracy	Kappa	Mean Absolute Error	FPR	Precision	Recall	ROC Area	Training Time
Naïve Bayes	55.5	0.4768	0.0893	0.013	0.763	0.555	0.896	1.76
REP Tree	83.23	0.7879	0.0335	0.023	0.833	0.832	0.905	7.92
Random Tree	86.46	0.827	0.0335	0.025	0.861	0.865	0.974	2.55
Random Forest	84.3	0.8018	0.0366	0.021	0.842	0.843	0.961	130.74
J48	86.17	0.8239	0.0323	0.023	0.859	0.862	0.961	39.86
Random Committee	84.2	0.8007	0.0353	0.02	0.842	0.842	0.925	21.98
Bagging	86	0.822	0.0348	0.024	0.862	0.86	0.977	60.92
MultiClass	81.18	0.7609	0.0378	0.035	0.805	0.812	0.913	12.66
Randomizable Filtered	75.93	0.6955	0.0481	0.04	0.76	0.769	0.86	5.4
IBK	78.59	0.7302	0.0428	0.027	0.788	0.786	0.88	12

Table 3: Classifier Evaluation Results using weka (dataset-2)

It is clear from table 3 that when we used dataset-2 using weka then the performance of Random Tree (86.46%), J48 (86.17%) and Bagging (86%) algorithm was almost same. But J48 (39.86 Sec) and Bagging (60.92 Sec) took more training time so again Random Tree (2.55 Sec) was the winner in performance. This time Random Forest took 130.74 seconds time in training so we can say that Random Forest is slower to train when we use larger datasets.

Classifier	Accuracy	Kappa	Mean Abs. Error	FPR	Precision	Recall	ROC Area	Training Time (Sec)
Naïve Bayes	55.44	0.4508	0.0887	0.022	0.786	0.554	0.882	0.4
REP TREE	91.61	0.8842	0.0214	0.01	0.916	0.916	0.995	1.2
Random Tree	93.01	0.9014	0.0295	0.008	0.93	0.93	0.996	0.45
Random Forest	91.88	0.8855	0.0252	0.009	0.921	0.919	0.994	3.2
J48	91.2	0.8758	0.0249	0.009	0.908	0.912	0.994	1.3
RandomCommittee	93.08	0.9024	0.0248	0.008	0.931	0.931	0.995	0.24
Bagging	91.53	0.8805	0.0294	0.01	0.914	0.915	0.992	1.11
MultiClass	92.6	0.8953	0.0274	0.011	0.924	0.926	0.996	0.9
Randomizable Filtered	90.76	0.8726	0.0263	0.01	0.9	0.908	0.993	0.88
IBK	90.31	0.877	0.0284	0.015	0.906	0.903	0.991	2.5

Table 4: Classifier Evaluation Results for dataset-1 using Spark (Dataset-1)

Other side, Naïve Bayes is easy to train as it took just 1.76 Seconds but its accuracy is not good in our case. Naïve Bayes perform well on classification of textual data but our dataset was having more features with numerical data.

It is clear from table 4 that when we used dataset-1 using spark then the Accuracy of Random Tree (93.01%) and Random Committee (93.08%) algorithms was equally high so both were the winners in Accuracy using spark.

Classifier	Accuracy	Kappa	Mean Abs. Error	FPR	Precision	Recall	ROC Area	Training Time (Sec)
Naïve Bayes	55.92	0.4803	0.0881	0.014	0.767	0.559	0.899	0.8
REP TREE	87.1	0.8351	0.0345	0.024	0.869	0.871	0.984	2.45
Random Tree	90.2	0.8754	0.0307	0.016	0.903	0.902	0.991	1.5
Random Forest	88.55	0.8559	0.0325	0.018	0.889	0.887	0.984	32
J48	87.93	0.8462	0.0319	0.021	0.881	0.879	0.986	12
RandomCommite	90.15	0.877	0.0284	0.015	0.906	0.903	0.991	4
Bagging	88.67	0.8559	0.0325	0.018	0.889	0.887	0.984	18
MultiClas	89.47	0.866	0.0281	0.015	0.9	0.895	0.988	7

Table 5: Classifier Evaluation Results for dataset-2 using Spark (Dataset-2)

It is clear from table 5 that when we used dataset-2 using spark then the Accuracy of Random Tree (93.20%) and Random Committee (93.15%) algorithms was equally high so both were the winners in Accuracy using spark.

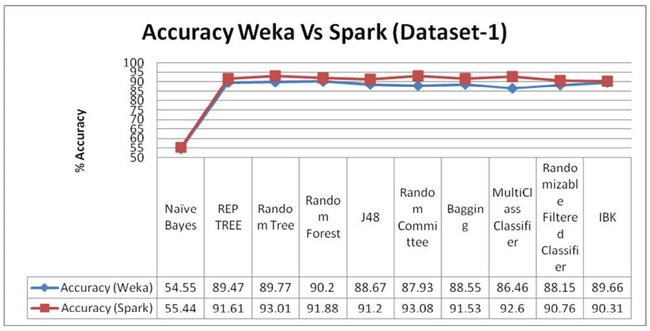
To measures for assessing how good or how "accurate" your classifieris at predicting the class label of tuples. The classifier evaluation measures are accuracy (also known as recognition rate), sensitivity (or recall), specificity, precision, F1.Note that although accuracy is a specific measure, the word "accuracy" is also used as a general term to refer to a classifier's predictive abilities. The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. That is,

$$\text{Accuracy}=\frac{TP+TN}{P+N}$$

Here, TP=True Positive, TN = True Negative, P=Positive, N=Negative.

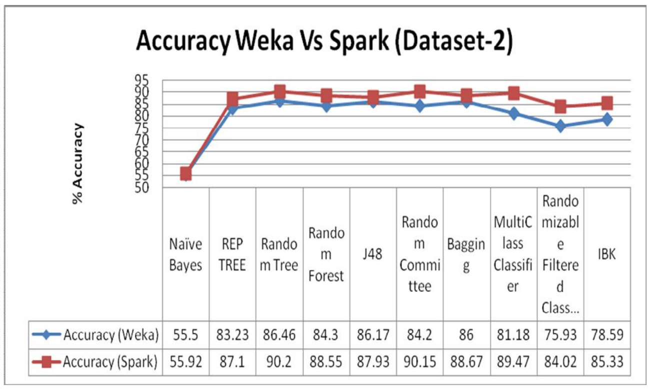
In the pattern recognition literature, this is also referred to as the overall recognition rate of the classifier, that is, it reflects how well the classifier recognizes tuples of the various classes.

<u>Comparison of Experimental results (Weka Vs Apache Spark) with graphs:</u>



	Naïve Bayes	REP TREE	Random Tree	Random Forest	J48	Random Committee	Bagging	MultiClass Classifier	Randomizable Filtered Classifier	IBK
Accuracy (Weka)	54.55	89.47	89.77	90.2	88.67	87.93	88.55	86.46	88.15	89.66
Accuracy (Spark)	55.44	91.61	93.01	91.88	91.2	93.08	91.53	92.6	90.76	90.31

Fig.3: Accuracy of Dataset-1 using Weka Vs Spark



	Naïve Bayes	REP TREE	Random Tree	Random Forest	J48	Random Committee	Bagging	MultiClass Classifier	Randomizable Filtered Class...	IBK
◆ Accuracy (Weka)	55.5	83.23	86.46	84.3	86.17	84.2	86	81.18	75.93	78.59
■ Accuracy (Spark)	55.92	87.1	90.2	88.55	87.93	90.15	88.67	89.47	84.02	85.33

Fig.4: Accuracy of Dataset-2 using Weka Vs Spark

V. CONCLUSION

We have compared the tools and algorithms discussed in this work and we have found that Apache spark and Random Tree are the winners based on various performance parameters.

Nowadays prevention of security breaches using the existing security technologies is unrealistic. As a result, intrusion detection is an important component in network security. Also, misuse detection technique cannot detect unknown attacks so the anomaly detection technique is used to identify these attacks. To improve the accuracy rate of intrusion detection in anomaly based detection data mining technique is used. In this paper, we have analyzed large datasets on Bigdata tool apache spark for anomaly detection using machine learning. The results of the analysis using basic weka tool are compared with the results using apache spark.

It is found that anomaly detection approach is more effective and fast on spark and random tree algorithm outperforms. The accuracy of random tree algorithm is better than other algorithms. This approach properly classifies the data either as normal and various attacks. Accuracy is also improved by using apache spark. Various filters were used to remove the meaningless, noisy and unrelated data from the original datasets. It can be concluded that this approach is better, faster and more efficient when used on apache spark.

REFERENCES

[1] Survey on Host and Network Based Intrusion Detection System, Niva Das, Tanmoy Sarkar, Int. J. Advanced Networking and Applications, 2014.

[2] Anomaly Based Intrusion Detection-A Review, Abhinav S. Raut, Kavita R. Singh, Int. J. on Network Security, Vol. 5, 2014.

[3] A Big Data Architecture for Large Scale Security Monitoring, Samuel Marchal_y, Xiuyan Jiangz, Radu State_, Thomas Engel, Springer, 2014.

[4] Big Data Analysis System Concept for Detecting Unknown Attacks, Sung-Hwan Ahn*, Nam-Uk Kim*, Tai-Myoung Chung**, IEEE, 2014.

[5] Autonomic Intrusion Detection System in Cloud Computing with Big Data, Kleber M.M. Vieira, Fernando Schubert, Guilherme A. Geronimo, Rafael de Souza Mendes, Carlos B. Westphall, Conference: The 2014 International Conference on Security and Management (SAM 2014).

[6] A data mining framework to analyze road accident data, Sachin Kumar1* and Durga Toshniwal2, Springer, 2015.

[7] Contextual anomaly detection framework for big sensor data, Michael A Hayes and Miriam AM Capretz*, Springer, 2015.

[8] Using Extreme Learning Machine for Intrusion Detection in a Big Data Environment, Junlong Xiang Magnus Westerlund Dušan Sovilj Göran Pulkkis, IEEE, 2014.

[9] Unsupervised Network Anomaly Detection in Real-Time on Big Data, Juliette Dromard, Gilles Roudi_ere, Philippe Owezarski, Springer, 2015.

[10] A Real-time Intrusion Detection System by Integrating Hadoop and Naive Bayes Classification, Sanjai Veetil, Dalhousie University, DSCI 2013.

[11] Hadoop-based Multi-classification Fusion for Intrusion Detection, Xun-Yi Ren and Yu-Zhu Qi, Journal of applied science 2013.

[12] Distributed Discord Discovery: Spark BasedAnomaly Detection in Time Series, Yafei Wu, Yongxin Zhu, Tian Huang, etc, IEEE, 2015.

[13] A Parallel Distributed Weka Framework for Big Data Mining using Spark, Aris-Kyriakos Koliopoulos, Paraskevas Yiapanis, Firat Tekiner, Goran Nenadic, John Keane. Ieee 2015.

[14] Statistical Technique for Online Anomaly Detection Using Spark Over Heterogeneous Data from Multi-source VMware Performance Data, Mohiuddin Solaimani, Mohammed Iftekhar, Latifur Khan, Bhavani Thuraisingham, IEEE, 2014.

[15] A hybrid approach for efficient anomaly detection using metaheuristic methods, Tamer F. Ghanem a,*, Wail S. Elkilani b, Hatem M. Abdul-kader, Journal of Advanced Research, 2014.

[16] UNSW-NB15: A Comprehensive Data set for Network Intrusion Detection systems, Nour Moustafa, Jill Slay, (IEEE),2015