Combining intrusion detection datasets using MapReduce

Conference Paper · October 2016

CITATIONS
25

READS
503

2 authors:

Mondher Essid
University of Sousse
2 PUBLICATIONS **29** CITATIONS
SEE PROFILE

Farah Jemili
University of Sousse
49 PUBLICATIONS **362** CITATIONS
SEE PROFILE

Combining intrusion detection datasets using MapReduce

Mondher Essid
ISITCOM Hammam Sousse,
University of Sousse, Tunisia
Email: mondher_ess@outlook.fr

Farah Jemili
ISITCOM Hammam Sousse,
University of Sousse, Tunisia
Email: Jmili_farah@yahoo.fr

Abstract —**Extensive use of computer network and huge amount of data had led several works to focus on intrusion detection system, which are based on single dataset or to combine multiple detection techniques to analyze detection rate and false positive flag.**

This paper presents a new way to combine intrusion detection dataset, we will concentrate about KDD99, DARPA dataset using bigdata technique (mapreduce). One main goal is to generate single dataset with different attack type that is realistic and meets real world criteria. Another major goal is to generate low false flag and higher detection rate.

This work consists in combining dataset and removing redundancy information using Bigdata technique, in final step we will implement NaiveBayes network and K2 algorithm using WEKA Tools to analyze the dataset.

Keywords— Bigdata, mapreduce, combine, intrusion detection dataset, KDD99, DARPA, NaiveBayes, K2.

I. INTRODUCTION

Intrusion detection often involves the analysis of Big Data. The Cloud Security Alliance reported that in 2013, a company like HP can generate one trillion events daily or about 12 million events per second [1]. It can be very difficult to correlate events over such large amount of data. To process this Big Data issues, existing work and research techniques based on analyzing each dataset one by one are not effective, also the rate of false alarms and detection rate are particularly problematic.

The remainder of the paper is organized as follow: Section 2 we will discuss about intrusion detection dataset. Next in section 3 we will explain the use of mapreduce. Section 4 we will provide methodology of the work. Results are reported in section 5 followed by a conclusion.

II. INTRUSION DETECTION

The KDD 1999 dataset used for benchmarking intrusion detection problems is used in our experiment [2]. The dataset was a collection of simulated raw TCP dump data over a period of nine weeks on a local area Network. The training data was processed about five million connections records for seven weeks of network traffic and two weeks of testing data yielded around two million connection records.

The training data is made up of 22 different attacks out of the 39 present in the test data. The known attack types are those present in the training dataset while the new attacks are the additional attacks in the test datasets which is not available in the training data sets. The attacks types are grouped into four categories [3]:

- Denial of Service Attack (DOS): denial of service attack is a class of attacks in which an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or deny legitimate users access to a machine.
- U2R: User to root exploits are a class of attacks in which an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain Root access to the system.
- R2L: A remote to user attack is a class of attacks in which an attacker sends packets to a machine over a network-but who does not have an account on that machine; exploits some vulnerability to gain local access as a user of that machine.
- Probing Attack (PROB): or Probing is a class of attacks in which an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits.

In each connection there are 41 attributes describing different features of the connection and a label assigned to each either as an attack type or as normal. Table 1 shows type of attack.

TABLE 1. TYPE OF INTRUSION DETECTION OF KDD99

Categories	**Specific Classification Identification**
DOS	Neptune, pod, land, back,smurf, teardrop
Probing	Ipsweep, nmap, portsweep, satan
R2L	Imap, ftp_write, Warezclient, multihop, phf, spy, guess_ passwd, warezmaster
U2R	Loadmodule, buffer_overflow, rootkit, perl

The protocols that are considered in KDD dataset are TCP, UDP, and ICMP that are explained below:

TCP: TCP stands for "Transmission Control Protocol". TCP is an important protocol of the Internet Protocol Suite at the Transport Layer which is the fourth layer of the OSI model. It is a reliable connection-oriented protocol which implies that data sent from one side is sure to reach the destination in the same order. TCP splits the data into labeled packets and sends them across the network. TCP is used for many protocols such as HTTP and Email Transfer.

UDP: UDP stands for "User Datagram Protocol". It is similar in behavior to TCP except that it is unreliable and connection-less protocol. As the data travels over unreliable media, the data may not reach in the same order, packets may be missing and duplication of packets is possible. This protocol is a transaction-oriented protocol which is useful in situations where delivery of data in certain time is more important than loosing few packets over the network.

ICMP: ICMP stands for "Internet Control Message Protocol". ICMP is basically used for communication between two connected computers. The main purpose of ICMP is to send messages over networked computers. The ICMP redirect the messages and it is used by routers to provide the up-to-date routing information to hosts, which initially have minimal routing information. When a host receives an ICMP redirect message, it will modify its routing table according to the message.

DARPA

The DARPA 1999 dataset consists of weeks one, two and three of training data and weeks four and five of testing data [4]. In training data, the weeks one and three consist of normal traffic and week two consists of labeled attacks.

In this work we are not interested in darpa payload information so we will only consider the first and third week of darpa dataset which we will combine it with KDD dataset. In order to combine those two datasets, we will use bigdata technique such us mapreduce which will allow us to resolve our problem.

III. BIG DATA

Hadoop meets the challenges of Big Data by simplifying the implementation of intensive data, highly parallel distributed applications [5]. Used throughout the world by businesses, universities, and other organizations, it allows analytical tasks to be divided into fragments of work and distributed over thousands of computers, hadoop provides a simple programming approach that abstracts the complexity evident in previous distributed implementations. As a result, Hadoop provides a powerful mechanism for data analytics, which consists of the following point [6]:

- *Vast amount of storage*

Hadoop enables applications to work with thousands of computers and petabytes of data. Over the past decade, computer professionals have realized that low-cost commodity systems can be used together with high-performance computing applications that once could be handled only by supercomputers. Hundreds of small computers may be configured in a cluster to obtain aggregate computing power that can exceed by far that of single supercomputer at a cheaper price. Hadoop can leverage clusters in excess of thousands of machines, providing huge storage and processing power at a price that an enterprise can afford.

- *Distributed processing with fast data access*

Hadoop clusters provide the capability to efficiently store vast amount of data while providing fast data access. This was because the cluster execution model creates demand for shared data storage with very high I/O performance. Hadoop moves execution toward the data. Moving the applications to the data alleviates many of the high performance challenges. In addition, Hadoop applications are typically organized in a way that they process data sequentially. This avoids random data access (disk seek operations), further decreasing I/O load.

- *Reliability, failover, and scalability*

In the past, implementers of parallel applications struggled to deal with the issue of reliability when it came to moving to a cluster of machines. Although the reliability of an individual machine is fairly high, the probability of grows as the size of the cluster grows. It will not be uncommon to have daily failures in a large (thousands of machines) cluster. Because of the way that Hadoop was designed and implemented, a failure (or set of failures) will not create inconsistent results. Hadoop detects failures and retries execution (by utilizing different nodes). Moreover, the scalability support built into Hadoop's implementation allows for seamlessly bringing additional(repaired) servers into a cluster, and leveraging them for both data storage and execution.

Hadoop development is driven by a goal to better support vast amount of data. Hadoop provides a powerful framework named MapReduce for easily writing application, furthermore providing highly scalable, parallelizable execution.

MapReduce

Hadoop MapReduce is a data processing framework that can be utilized to process massive amount of data stored in HDFS (It's a block structured distributed file system that is designed to store petabytes of data) distributed processing of a massive amount of data in a reliable and efficient manner is not an easy task [7]. Hadoop MapReduce aims to make it easy for users by providing a clean abstraction for programmers by providing automatic parallelization of the programs and by providing framework managed fault tolerance support.

MapReduce programming model consists of Map and Reduce functions. The Map function receives each record of the input data (lines of a file, rows of a database, and so on) as key-value pairs and outputs key-value pairs as the result. By design, each Map function invocation is independent of each

other allowing the framework to use divide and conquer to execute the computation in parallel.
This also allows duplicate executions or re-executions of the Map tasks in case of failures or load imbalances without affecting the results of the computation. Hadoop MapReduce groups the output key-value records of all the Map tasks of a computation by the key and distributes them to the Reduce tasks. This distribution and transmission of data to the Reduce tasks is called the Shuffle phase of the MapReduce computation. Input data to each Reduce task would also be sorted and grouped by the key. The Reduce function gets invoked for each key and the group of values of that key (*reduce<key, list_of_values>*) in the sorted order of the keys. In a typical MapReduce program, users only have to implement the Map and Reduce functions and Hadoop takes care of scheduling and executing them in parallel.
In this work we will focus on Mapreduce framework, which will allow us to combine datasets.

IV. CONTRIBUTION

In order to combine our two datasets our work is divided in three major step:

A. The first step to remove the redundancy instance from both dataset KDD and DARPA.
B. The second step is horizontal combination of both instance files of the dataset.
C. Finally, we will combine all files in a vertical way.

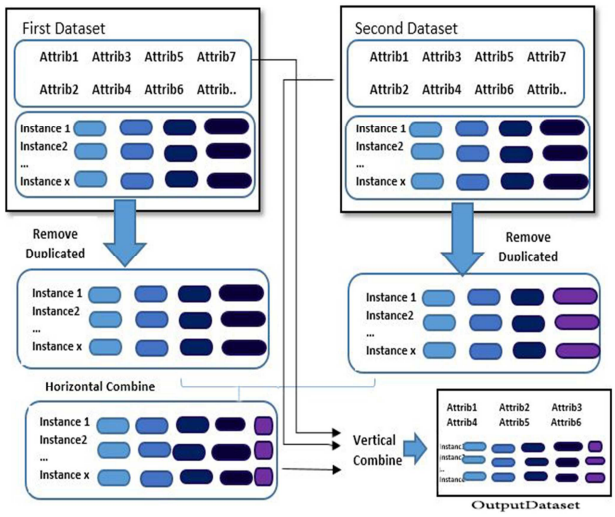

Figure 1. The main idea of fusion model

A. *Remove redundancy*

In order to combine the two datasets our first step is to remove duplicate instances data lines from both KDD and DARPA that will help us to extract only the useful information. The desired output should contain all of the input values, but values that appear multiple times in input, must appear only once in output.

The solution is simpler than that for word count: we use the value itself as the map output key; the reducer associated with a specific key, can then return a *single* output.

- **Map**: For each input value *v*, output the key-value pair (*v*, *v*)
- **Shuffle**: For each key *v* produced by any of the Map tasks, there will be one or more key-value pairs (*v*,*v*). Then, for key *v*, the shuffle will produce (*v*, [*v*, *v*, ..., *v*]) and present this as input to a reducer.
- **Reduce**: the reducer for key *v* turns (*v*, [*v*, *v*, ..., *v*]) into *v*, so it produces exactly one output *v* for this key *v*.

After removing redundancy instances data lines from both datasets we will move to next step which allow us to combine both instance file of KDD and darpa dataset.

B. *Horizontal combine*

Joins is one of the interesting features available in MapReduce.
The Reduce-Side join seems like the natural way to join datasets using Map/Reduce [8]. It uses the framework's built-in capability to sort the intermediate key-value pairs before they reach the Reducer. But this sorting is often a very time consuming step. So we used another join technique.

MapSidejoin:

Hadoop offers another way of joining datasets before they reach the Mapper. This functionality is present out of the box and is arguably the fastest way to join two datasets and have many advantages like:

- Reducing the expenses that we use for sorting and merging in the shuffle and reduce stages.
- Optimizes the performance of the task i.e. reducing the time to complete the task.

Mapside join places constraints on the datasets that can be used for the join, the main two conditions are:

- The both datasets to be joined are already sorted and have the same number of partitions.
- One of the two datasets to be joined, one is small enough to fit into memory

In our case we respond to both conditions. The two datasets have the same number of instances and our dataset can be fit into memory. When running Mapreduce job will simply perform one to one join by mapping each instance from the first file with the first instance of the second dataset, after performing all dataset we will get our combined dataset in the correct form. The final step is to achieve vertical combine of both attributes files and instance file.

C. *Vertical combine*

We will merge our three files into sequence file using custom record reader and custom input format classes by extending InputFormat and Record Reader classes from Hadoop API.

Each file will be processed as a single record and we need those two classes to perform this job.

The file is read as input from HDFS directory to a single file in output HDFS directory. So we are using single reducer to produce a single output sequence file.

In order to prevent file splitting during the map phase we are overriding isSpittable() method and returning false and we are overriding createRecordReader () to return custom Record Reader instance.

V. EXPERIMENTATION

In our experimental result we will use WEKA toolkit to analyze the dataset with NaiveBayes and K2 algorithm.

Bayesian Networks

After fusion the dataset we are performing the intrusion detection based on Bayesian network. It is the probabilistic approach to detect the intrusion.

Bayesian Networks model probabilistic relations among random variables [9]. A random variable is a term from Statistics that means the items involved vary in some random, or unexplained manner. A Bayesian Network assigns probability factors to various results based upon an analysis of a set of input data. Like many other Machine Learning algorithms, a Bayesian Network is taught using training data.

K2 Algorithm

K2 learning algorithm showed high performance in many research works.

K2 algorithm is one of the special algorithms used for performance improvement in Bayesian network, and a technique that can effectively optimize each node.

K2 algorithm proceeds by starting with a single node (the first variable in the defined order) and then incrementally adds connection with other nodes which can increase the whole probability of network structure [10].

WEKA

WEKA is an open source application formally called Waikato Environment for Knowledge Learning [11], is a computer program that was developed at the University of Waikato in New Zealand for the purpose of identifying information from raw data gathered from agricultural domains. It supports many different standard data mining tasks such as data pre-processing, classification, clustering, regression, visualization and feature selection. The basic premise* of the application is to utilize a computer application that can be trained to perform machine learning capabilities and derive useful information in the form of trends and patterns. WEKA operates on the predication that the user data is available as a flat file or relation, this means that each data object is described by a fixed number of attributes that usually are of a specific type, normal alpha-numeric or numeric values. The WEKA application allows novice users a tool to identify hidden information from database and file systems with simple to use options and visual interfaces

The main criteria that we have considered in this work are the detection rate, false positive:

- *TP Rate:* rate of true positives or detection rate (instances correctly classified as a given class)

TP = (Total_detected _attacks/Total_attacks) *100

- FP Rate: rate of false positives (instances falsely classified as a given class)

FP = (Total_misclassified _process/Total_normal_attacks) *100

In this section, we will compare the performance of the proposed intrusion detection with the experimental results of F. JEMILI, M. ZAGHDOUD, M. BEN AHMED [12] paper which are taken for a comparative study. Table 2 gives the detection rate of the proposed system and the F. JEMILI, M. ZAGHDOUD, M. BEN AHMED [12] work. The detection rate for DOS is 88% in previous work [12] and it is 99% for the proposed system. Similarly, for U2R, and R2L, Normal, there is a high improvement in detection rate while comparing with previous work [12].

The low performance of our system in detection of SSH connections may be explained by the low proportions of SSH training connections.

TABLE 2. COMPARISON OF DETECTION RATE

Intrusion	**Detection Rate**	
	F. JEMILI, M. ZAGHDOUD, M. BEN AHMED [12]	***Proposed system***
DOS	88.64%	99.96%
Probing	99.15%	97.02%
U2R	6.66%	93.32%
R2L	20.88%	97.41%
Normal	87.68%	97.40%
SSH	-	57.1%
NOTSSH	-	82.5%

TABLE 3. FALSE POSITIVE RESULT IN TRAINING DATE SET

Intrusion	Proposed Method
DOS	0%
Probing	0.25%
U2R	0.1%
R2L	0.13%

The classification performance of IDS is measured by false alarm rate too, table 3 shows high performance of our system in false positive result. The average false alarm rate achieved is 0.58%.

Conclusion

In this paper, we have presented a new approach based on mapreduce which allowed us to combine both dataset KDD and DARPA. Then we implemented BayesienNetwork and K2 algorithm to analyze our dataset. The main merits of our work on intrusion detection are two folds. The first is to initiate a new way to combine IDS dataset and the second is to show the power of MapReduce implementation to combine and handle large amount of training data.

Our work shows a good performance in detection phase and low false alarm. This is due to the use of Bayesian networks.

However, there is another problem to be solved in our approach, it's the ability to handle heterogeneous dataset with different dataset structures.

In the future work, we will further develop our approach in order to handle different dataset structures. We will try also to make improvement on performance of detection rate and predicting attacks

REFERENCES

[1] Group BDW Big Data Analytics for Security Intelligence. Accessed 2015–1-10. https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Analytics_for_Security_Intelligence.pdf

[2] KDD Cup 1999 Data: http://kdd.ics.uci.edu/databases/kddcup99

[3] M Ajith Abraham, M Mario Koppen "Intelligent Systems Design and Applications" pp.240-245

[4] M Erland Jonson, M Alfonso Valdes, M Magnus Almgren "Recent Advance in Intrusion Detection" pp.215

[5] M Boris Lublinsky, M Kevin T. Smith, M Alexey Yakubovich "Professional Hadoop Solutions" pp.5-6

[6] M Boris Lublinsky, M Kevin T. Smith, M Alexey Yakubovich "Professional Hadoop Solutions" pp.30

[7] M ThilinaGunarathne "Hadoop MapReduce v2 Cookbook - Second Edition" pp.12

[8] MapJoinOptimization [online]. Avaible:http://wiki.apache.org/hadoop/Hive/JoinOptimization

[9] J. Pearl, "Probabilistic Reasoning in Intelligent Systems:Network of Plausible Inference". Morgan Kaufmann,1997.

[10] Jorge E.Hernandez Pascale Zarate FatimeDargam "Decision Support Systems- Collaborative Models and Approaches in Real Environments" pp.61

[11] The WEKA data mining software: An update, Mark Hall, Eibe Frank, G. Holmes, B. Pfahringer, P. Reutemann, IH Witten, ACM SIGKDD Explorations, Newsletter, Pages 10-18, volume 11 issue 1, June 2009

[12] F. JEMILI, M. ZAGHDOUD, M. BEN AHMED. "A Framework for an Adaptive Intrusion Detection System using Bayesian Network". In proceedings of the IEEE International Conference on Intelligence and security informatics, NEW BRUNSWICK/NJ, USA, 2007.