Using MongoDB Databases for Training and Combining Intrusion Detection Datasets

Marwa Elayni and Farah Jemili

Abstract A single source of intrusion detection dataset involves the analyze of Big Data, recent attempts focus on Big Data techniques in order to combine heterogeneous data sets and solve the problems of analyzing the huge amounts of data. The main objective of this paper is to present a method to train and combine several datasets from semi-structured sources with the MapReduce programming paradigm under MongoDB. It aims to increase the intrusion detection rates. In our work, we will focus on KDD99, DARPA 1998 and DARPA 1999 dataset and with the big data technique MapReduce in MongoDB: First, we will select the most pertinent attributes and eliminate the redundancies from the previous datasets. Then, we will merge them vertically into the same collection. Finally, to analyze the dataset we will use a Bayesian network as K2 algorithm implemented in WEKA.

Keywords Intrusion detection · Big Heterogeneous Data · NOSQL system · MapReduce · MongoDB · KDD99 · DARPA · K2

1 Introduction

All intrusion detection systems involve an intrusion detection dataset, a learning system and an inference system. Most existing researches deploy a single intrusion detection dataset for system learning and inference for which these works use several classification methods such as Bayes networks, neural networks, fuzzy neural network and genetic algorithms [1–4].

The results of the intrusion detection of these works do not show high performance at intrusion detection rate, so the idea is to invest within the intrusion detection datasets to increase the intrusion detection rates.

M. Elayni (✉) · F. Jemili (✉)
ISITCOM Hammam Sousse, University of Sousse, Sousse, Tunisia
e-mail: aynimarwa@gmail.com

F. Jemili
e-mail: Jmili_farah@yahoo.fr

R. Lee (ed.), *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Studies in Computational Intelligence 721,
DOI 10.1007/978-3-319-62048-0_2

In literature we find many of intrusion detection datasets such as KDD 99, NSL KDD, DARPA, CAIDA, ADFA,… these datasets are characterized by a huge data size, unstructured format. Then a use of Big Data techniques is essential in the intrusion detection. And now due to Big Data, we are able to manage and treat a huge size of instances stored in different intrusion detection datasets.

In this context, we cite the example of the commercial SIEM (Security Information and Event Management) [5, 6] using relational database technologies for storage repositories, found that databases has become bottlenecks in deployments at larger enterprises: storage and retrieval of data start to taken more time which is unacceptable.

It is a clear that SIEM data, presenting the intrusion detection traffic, are facing Big Data challenges where as the relational databases are becoming bottlenecks.

So, next generation Big Data storage technologies like NoSQL databases can help in addressing these problems.

The purpose of this paper is to describe a training model for three datasets and to fuse those datasets in a single dataset with NoSQL database MongoDB to achieve the goal which is obtaining higher intrusion detection rates and lower false alarms.

We started by introducing the work. The second section describes the intrusion detection datasets used in our model. In the third section, we justify our choice of MongoDB as NoSQL system. In the fourth section we explain the use of MongoDB. In the fifth section, we provide experimental results.

2 Intrusion Detection Datasets

In our work we use three datasets KDD99, DARPA98 and DARPA99, our choice is based on a study by Azad and Jha [7] published in a journal "the intrusion detection and Big Heterogeneous Data" [6]: 46 out of 75 studies used either DARPA or KDD Cup while only 29 chose a different one.

Although these two dataset have been there for over a decade, they are still considered as the two most popular datasets used for intrusion detection researches. Even in our approach, we used the following datasets:

KDD99

The KDD Cup 1999 dataset which is used in our experiment [8], used for benchmarking intrusion detection problem. The dataset is a collection of simulated raw TCP dump data during a period of nine weeks on a local network area: seven weeks of network traffic that gives about five million connections records of the training data and two weeks of testing data is giving around two million connections records.

DARPA

DARPA99 and DARPA 98 traces are generated by the MIT Lincoln Labs for intrusion detection evaluation [9, 10]. The DARPA98 traces are consisted of: training data seven weeks and testing data two weeks.

Table 1 The Name of some attacks of each category _KDD, DARPA

Categories	Name of attack
Probing	Ipsweep, nmap, portsweep, satan, IPsweep, saintmscan, nmap, …
DOS	Neptune, pod, land, back, smurf, teardrop, …
U2R	Loadmodule, buffer_overflow, rootkit, perl, format, PS, …
R2L	Imap, ftp_write, Warezclient, multihop, phf, spy, guess_passwd, warezmaster, imap, worm, …

The DARPA 99 is consisted of five weeks: the first three weeks are dedicated for training data, while the last two weeks are for testing and that gives about six million connections records.

For each connection present in KDD and DARPA, there are attributes which are dedicated for each dataset, these attributes describe different features of the connection and contain a specific label assigned to indicate the type of label and whether it's an attack or normal one. There is a large variety of attacks; most of them are grouped into one of the following categories [11]:

- A Probe which is an attempt to learn information that could facilitate an attack.
- A Denial of Service (DoS) which is an attack that overloads the resources of a system and aims to make the services or the resources of an organization unavailable during an indefinite time.
- A user to root (U2R) which is an attack where a user with limited right access attempts to gain root permissions.
- A remote to user (R2L) it's the case when a user, that is unknown to the system, attempt to own the user permissions in order to expose a machine vulnerabilities via internet.

The Table 1 shows some attacks present in KDD and DARPA and for each attack we indicate its category.

In order to present our model for training and fusion those three datasets, we used NoSQL technique, in the next section; we define the NoSQL systems, ending in approving our choice of the best efficient NoSQL systems.

3 NoSQL Databases

Since 2012 the data volume has evolved from a few dozen terabytes to many petabytes [12], so necessary techniques are mandatory to capture, manage and process these huge amounts of datasets "Big Data" within a tolerable elapsed time.

Currently, there are several solutions for the analysis of big data that can be mentioned: Search-based systems, New SQL and NoSQL databases [12]. We are interested in NoSQL databases for resolving the problem of analytic data for intrusion detection system.

Table 2 Types and examples of NOSQL databases

Types	Examples
Key/value databases	Couchbase, Dynamo, redis, Riak, orientDB…
Document store	MongoDB, Apache CouchDB…
Column-oriented	Cassandra, HBase, vertica…
Graph	Neo4J, Allegro Graph, stardog…

There are many types of data storage models in NoSQL databases that are classified into four categories. In Table 2 we present examples of NoSQL databases for each category.

In this context, authors in [12] have compared eight NoSQL databases and two promising New SQL databases based on the following criteria's: performance, reliability, integrity, security, query complexity, interoperability and cloud support.

Basing on this work, we choose the most efficient NoSQL system. It calculates the global score of each system, the comparative study start by calculating the weight of each criteria (performance, reliability, integrity, security and query complexity, interoperability and cloud support) by ROC (rank order centroid) method.

The result of ROC method is (0.37 for performance, 0.23 for integrity, 0.16 for reliability, 0.11 for interoperability, 0.07 for cloud support, 0.04 for query complexity, 0.02 for security). Then the authors calculates the global score, by attributing a note for each criteria for each NoSQL system.

The global score calculation formula is:

$$\begin{aligned}\textbf{Globalscore} = {} & (\textbf{0.37}\,\text{performance}) + (\textbf{0.23} * \text{integrity}) + (\textbf{0.16} * \text{reliability}) \\ & + (\textbf{0.11} * \text{interoperability}) + (\textbf{0.07} * \text{cloud support}) \\ & + (\textbf{0.04} * \text{query 4 complexity}) + (\textbf{0.02} * \text{security})\end{aligned}$$

We focus only on the global score of the NoSQL systems, the results of the comparison are illustrated in the Fig. 1.

We focus only on the global score of the NoSQL systems, the results of the comparison of the global score shows that MongoDB system has obtained the best global score (4.96) [12].

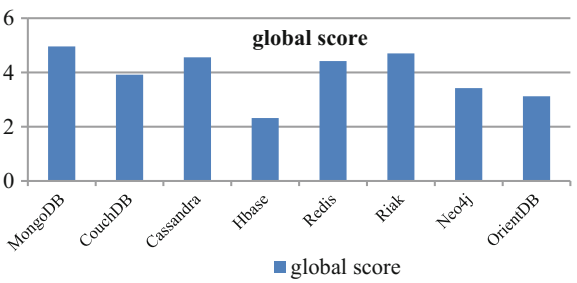

Fig. 1 The global score for the NoSQL systems

MongoDB, developed since 2007 by the 10gen software company, is a database management system that orients document written in C++ and very suitable for web applications, and MongoDB are the most popular NoSQL database [13].

In MongoDB, there are many techniques to achieve our model like indexing, aggregation. But the purpose in this paper is to present a model able to train and combining a real size of traffic of intrusion detection. So we decided to use a distributed programming technique MapReduce which is capable of managing large quantities of data [14, 15].

MapReduce is a programming model and this framework is an associated implementation for processing huge amounts of datasets.

MapReduce can deal with a large dataset by dividing in several tasks. It is constituted by two necessary functions, the first one (Map) is associated for emitting a key/value pair to generate a set of intermediate key/value pairs for each task of the dataset. And the second one (Reduce) merges all intermediate values, emitted from all tasks, related with the same intermediate key. The Reduce functions start only when all the map functions are finished.

In the next section, we present our model for training and combining semi-structured instances from the numerous sources based on the MapReduce framework in MongoDB.

4 Proposed Method

In order to train and combine the three datasets, our work is divided into two major steps:

A. The first one is to remove the redundancies instances and select the most pertinent features from triple datasets KDD99, DARPA99 and DARPA98.
B. The second is a vertical combination of triple instance files of the dataset (Fig. 2).

In our method, we presented a model so the size of those datasets does not reflect the real size of traffic of intrusion detection. Also, our model is able to manage heterogeneous format of data like son, picture. We describe our method in the following parts.

4.1 Pre-processing of the Dataset

Feature Selection

Intrusion detection system deals with huge amount of data which contains irrelevant and redundant features. Based on many works obtained, we have selected the most pertinent attributes and eliminate those that carry no information or information redundant.

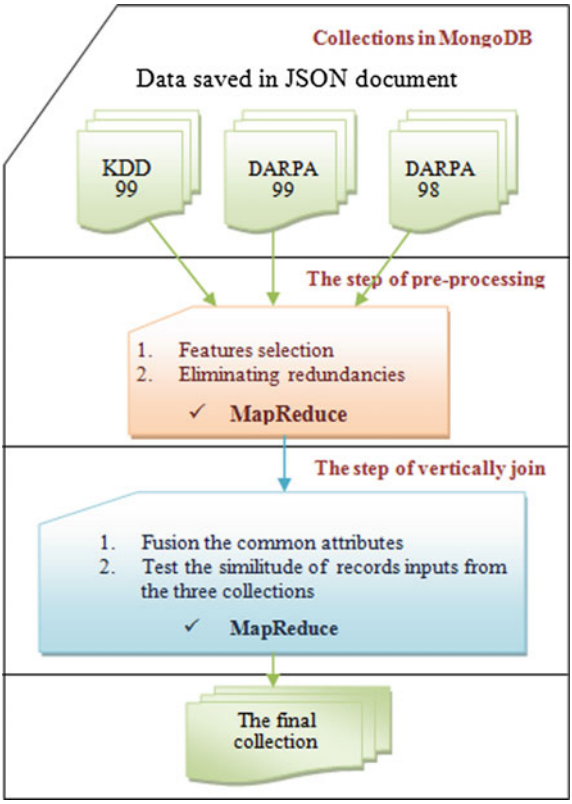

Fig. 2 Proposed method

We selected the most pertinent attributes in KDD that, according to [16, 17], we have used the method AFCM analysis (Factorial Multiple Correspondence Analysis) for this selection and choose those having the best gain. The attributes selected are: count, src_bytes, src_count, service, dst_host_same_src_port_rate, protocol_type, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_same_srv_rate.

In darpa 98, we selected only the useful features and eliminate information either redundant or not useful that identify the timestamp, source host and port, destination host and port [18].

Also, in darpa 99 and according to [19], the selection procedure of the features is managed by the ML algorithm. The attributes selected are: minfpktl, meanfpktl, maxfpktl, stdfpktl, minbpktl, protocol, fpacket.

The purpose of eliminating no-useful attributes is to attain a suitable classification for the evaluation of intrusion detection system and a better performance from the system.

Eliminating Redundancies

We should mention that in KDD 99, DARPA98 and DARPA 99, there are a large number of redundant recordings that causes a problem in terms of learning

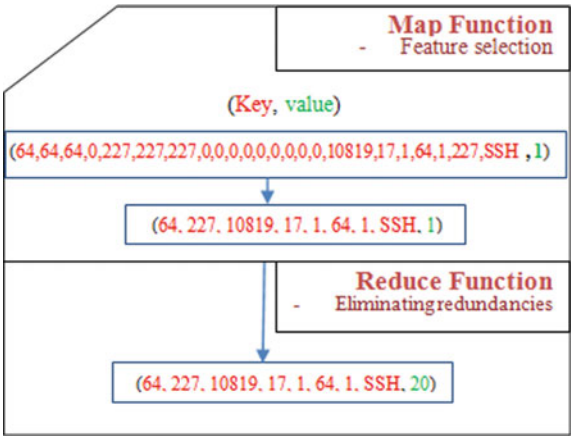

Fig. 3 The step of pre-processing with example from darpa99

algorithm and because of this redundancy, the learning algorithm biased toward the frequent records such as DOS and Probe attacks and consequently prevent them from learning the infrequent records such as U2R and R2L that are considered more noxious to the network.

In this step, the inputs of each data in MongoDB are stored in a different collection. Each line of the input data represents a connection and is saved as a document in the JSON format. And to achieve the step of pre-processing, we used a MapReduce under MongoDB for the two steps of pre-processing which we apply the procedure of eliminating redundancies just for the pertinent features (Fig. 3).

- **Map**: we applied a map function for each document in the collection. We emit for each document a pair:
  - **key** (the value of feature selection).
  - **value** (attribute 1 for each document to indicate that this document exists only once).
- **Reduce**: the Reduce function count the number of redundancies of each pair (key, value) emitted from function map by merging all intermediate values associated with the same intermediate key.
- The result is saved under the new collection.

So, in the map function, we selected the most pertinent attributes and in the reduce function, we eliminate the redundancy.

After we wrap up with the step of selecting the most pertinent attributes and removing the redundancies, we moved on to the next step to present a method with MapReduce in MongoDB to combine our datasets.

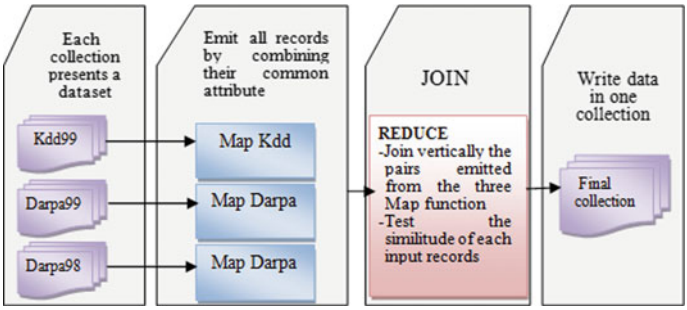

Fig. 4 Vertical combine of the three datasets in the same collection

4.2 Vertical Combine

Now, we have three datasets without redundancy. It just remains to merge the triple into a single dataset; the idea is described in the Fig. 4.

In this step our goal is to join vertically the three collections in MongoDB in one collection by fusion their common attributes. So the idea is to write a three map functions and only one reduce function.

- **Map Kdd, Map Darpa98 and Map Darpa99**: Each map function emit for each document:
  - key (name of each attribute by combining their common attribute).
  - value of each attribute (Fig. 5).
- **Reduce**: for each (key, value) sent by the three functions (map_kdd, map_-darpa98 and map_darpa99); A reduce function test the similitude of the values emitted from map functions by combining the similar records vertically for given more precision to each record (Fig. 6).

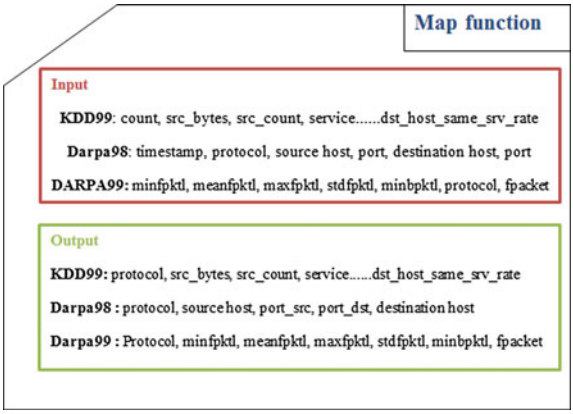

Fig. 5 The principle of combining the common attribute from the three datasets

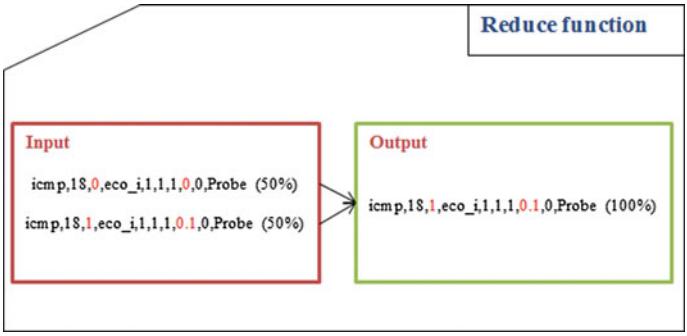

Fig. 6 The principle of the step of testing the similar records

- This result of all inputs records is saved vertically under the same new collection.

This fusion gives more diversity of attacks type, also gives more precision to each record by testing the similitude of each input record. All this improvement made necessary a big modification in intrusion detection rate. It increased the true positive rate and low down the false alarm rate.

Finally, when exporting the final collection, we used an aggregation to correct the problem of exporting a field from a subdocument.

5 Experiment and Results

In experimental result, we used the WEKA toolkit to analyse the dataset.

WEKA

Waikato Environment for Knowledge Analysis [20] is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. This toolkit contains a collection of algorithms and visualization tools for data analysis and predictive modeling, with graphical user interfaces for easy access to these functions.

We are interested in a classify panel, which enables the application off classification and regression algorithms to the resulting dataset, to estimate the accuracy of the resulting predictive model (TP Rate), and to visualize erroneous predictions (FP Rate), receiver operating characteristic (ROC) curves, etc.

There are many algorithms to classify a dataset like Bayesian networks, naive Bayes, decision tree, etc.

For our experimentation we used the Bayesian Networks as K2 algorithm. We used this one to calculate the rate of intrusion detection (TP rate) and also the rate of false alarm (FP rate).

The K2 algorithm is described as follows:

K2 Algorithm
K2 learning algorithm is an algorithm with superior quality; it is specialized and useful for performance improvement in Bayesian network and a technique to optimize each node. The procedure of K2 algorithm is begins by a single node and its increment to connect others nodes for increasing the probability of network structure [21].

The following formula used to measure the performance:

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

$$TP = (\text{Total detected } \_\text{attacks}/\text{Total\_attacks}) * 100$$

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

$$FP = (\text{Total\_misclassified\_process}/\text{Total\_normal\_attacks}) * 100$$

The deployment of Big Data technologies in the domain of intrusion detection is a new method. Recently, the work of Essid and Jemili [22] was published, their method serves to horizontally combine two sources of datasets (kdd99 and darpa 99) using the Big Data technique Map Reduce in Hadoop and they used the K2 algorithm, as a Bayesian network, for analysing their output.

In our work, we used the K2 algorithm, so to evaluate our work; we compared the results of our method with two methods using the same algorithm (K2):

- The method of Essid and Jemili [22].
- The method of Jemili et al. [23], they used a single dataset (kdd99) dataset.

When we compare the experimental results of Big Data technologies (MongoDB, Hadoop) with the work of Jemili et al. [23] in Table 3, we notice a remarkable amelioration of both MongoDB and Hadoop in term of detection rates.

For the results in Table 3 of MongoDB and the result of Hadoop, we can see that the two systems gives a good performance and we notice that MongoDB gives a

Table 3 Comparing the results of detection rates

Name of attack	Results with single dataset [23] (%)	Results with MongoDB (%)	Results of [22] Hadoop (%)
Normal	**87.68**	**99.80**	**97.40**
DOS	**88.64**	**98.40**	**99.96**
Probe	**99.15**	**99.90**	**97.02**
U2R	**6.66**	**88.90**	**93.32**
R2L	**20.88**	**98.10**	**97.41**
SSH	**–**	**65.30**	**57.1**
NOTSSH	**–**	**96.20**	**82.5**

Table 4 False positive in our proposed method

Types of attack	False positive (%)
DOS	0
Probe	0.2
U2R	0
R2L	0.2

certain amelioration better then Hadoop in term of results of detection rates in connections: Normal (our method gives 99.80% and Hadoop 97.40%), Probe, also for R2L (from 97.41 to 98.10%), SSH and NOTSSH. And it's necessary to indicate the number of connections for the both model Hadoop and MongoDB:

- For the output of Hadoop considered about 900,000 connections with size 120 MB.
- For the output of MongoDB considered about 600,000 connections with size 50 MB.

Although the difference between size for the both model, our model gives a better result then Hadoop and this improvement method is due to the step of selecting features when we selected the most pertinent features, the step of eliminate the redundancies from this pertinent features and the step of testing of similitude of connections emitted from the different sources, on the contrary Essid and Jemili [22] used all features for the two datasets.

For the two connections DOS and U2R, the method of Hadoop gives the results of detection rates better than our method with MongoDB. This improvement of method with Hadoop is due to the frequent number of connections of DOS and U2R.

Finally, we present the performance of our model in false positive results in the Table 4 and it's clear that our method gives high performance in terms of false positive.

So, the idea of using the Big Data storage as MongoDB can help in analyzing the traffic of intrusion detection, it would only take about few minutes to achieve the pre-processing of three datasets, also our model able to combining heterogeneous dataset like song or picture and other unstructured data. So the merging of different kind of datasets improves the results of performance indicators and subsequently the security system.

6 Conclusion and Future Work

The biggest challenge in the intrusion detection systems is the Big Data that are associated with large amounts of network traffic collected dynamically in the intrusion detection.

So, the big data techniques can necessary help the intrusion detection system for the management and the storage of several datasets and it is a new way for researchers to generate their own intrusion detection dataset.

In this paper, we used a NoSQL technology as MapReduce in MongoDB for the training and merging of different datasets kdd99, darpa1998 and darpa1999. Then, for the analyse of our dataset and the calculation of the performance metrics of our method, we used the K2 algorithm, as a Bayesian network, in order to compare it with other results: Our work showed a much better result than using a single dataset and better than Hadoop in some categories of attacks as Normal, Probe, R2L, SSH, NOTSSH. This compilation of big data techniques and intrusion detection system is greater and can necessary ameliorates the domain of security.

In future work, we will continue to develop our approach to merge others different distributed datasets by integrating MongoDB with others Big Data techniques and we will try to ameliorate the performance of detection rates and low false alarms. Also, we will integrate the MapReduce technique for developing distributed algorithms that would be able to process the new model of data.

References

1. Shanmugavadivu, R., Nagarajan, N.: Network intrusion detection system using fuzzy logic. Indian J. Comput. Sci. Eng. (IJCSE) **2**(1), 101–111 (2011)
2. Zekri, M., Meslati, L.S.: Immunological approach for intrusion detection. ARIMA J. **17**, 221–240 (2014)
3. Nadjaran Toosi, A., Kahani, M., Monsefi, R.: Network intrusion detection based on neuro-fuzzy classification. In: International Conference on Computing and Informatics, 2006. ICOCI '06, Kuala Lumpur, June 2006 (2006)
4. Hoque, M.S., Mukit, M.A., Bikas, A.N.: An implementation of intrusion detection system using genetic algorithm. Int. J. Netw. Secur. Appl. (IJNSA) **4**(2), 109–120 (2012)
5. Chickowski, E.: A case study in security big data analysis. http://www.darkreading.com/analytics/security-monitoring/a-case-study-in-security-big-data-analysis/d/d-id/1137299 (2012). Accessed Oct 2016
6. Richard, Z.T.M.Kh., Wald, R.: Intrusions detection and big heterogeneous data: a survey. J. Big Data 1, Article 115 (2015)
7. Azad, C., Jha, V.K.: Data mining in intrusion detection: a comparative study of methods, types and data sets. Int. J. Inf. Technol. Comput. Sci. (IJITCS) 75–90 (2013)
8. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/. Accessed mars 2016
9. DARPA1998 Data. https://www.ll.mit.edu/ideval/data/1998data.html. Accessed mars 2016
10. DARPA1999 DATA. https://web.cs.dal.ca/~riyad/Site/Download.html. Accessed mars 2016
11. Lee, C.: An evaluation of machine learning techniques in intrusions detection. Doctoral Thesis in Computer Science, University of Vanderbilt, P6 (2007)
12. Dehbi, O.R., Talea, M., Batouta, Z.I.: An advanced comparative study of the most promising NoSQL and NewSQL databases with a multi-criteria analysis method. J. Theoret. Appl. Inf. Technol. **81**(3) (2015)
13. DB-Engines Ranking. http://db-engines.com/en/ranking. Accessed May 2016
14. MongoDB database. https://docs.mongodb.com/manual/core/map-reduce/. Accessed Sept 2016

15. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, Dec 2004
16. Wei, W., Gombault, S., Guyet, T.: Towards fast detecting intrusions: using key attributes of network traffic. In: The Third International Conference on Internet Monitoring and Protection, Bucharest, vol. 13, pp. 86–91 (2008)
17. Al-Mamory, S.O., Jassim, F.S.: Evaluation of different data mining algorithms with KDD CUP 99 Data Set. J. Babylon. **21**, 2663–2681 (2013)
18. Zargar, G., Kabiri, P.: Identification of effective network features to detect Smurf attacks. In: Research and Development, SCOReD, p. 185 (2009)
19. Szabo, G.: Methods for efficient classification of network traffic. Thesis, Budapest University of Technology and Economics, p. 10 (2010)
20. Introduction to Weka. http://www.iasri.res.in/ebook/win_school_aa/notes/WEKA.pdf. Accessed 2016
21. Hernandez, J., Zarate, P., Dargam, F.: Decision Support Systems—Collaborative Models and Approaches in Real Environments, p. 61 (2011)
22. Jemili, F., Essid, M.: Combining intrusion detection datasets using MapReduce. In: The International Conference on Systems, Man, and Cybernetics (2016)
23. Jemili, F., Zaghdoud, M., Ahmed, M.B.: A framework for an adaptive intrusion detection system using Bayesian network. In: The IEEE International Conference on Intelligence and Security Informatics, USA, 2007 (2007)

http://www.springer.com/978-3-319-62047-3

Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing

Lee, R. (Ed.)
2018, XI, 215 p. 83 illus., 39 illus. in color., Hardcover
ISBN: 978-3-319-62047-3