computers

MDPI

Article

Network Intrusion Detection with a Hashing Based Apriori Algorithm Using Hadoop MapReduce

Nureni Ayofe Azeez [1], Tolulope Jide Ayemobola [1], Sanjay Misra [2,3], Rytis Maskeliūnas [4] and Robertas Damaševičius [4,*]

1 Department of Computer Sciences, University of Lagos, Lagos 100213, Nigeria; nazeez@unilag.edu.ng (N.A.A.); ayemobolatolulope@gmail.com (T.J.A.)
2 Department of Electrical & Information Engineering, Covenant University, Ota 112233, Nigeria; sanjay.misra@covenantuniversity.edu.ng
3 Department of Computer Engineering, Atilim University, Ankara 06830, Turkey
4 Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland; rytis.maskeliunas@polsl.pl
* Correspondence: robertas.damasevicius@polsl.pl

Received: 23 October 2019; Accepted: 29 November 2019; Published: 2 December 2019

Abstract: Ubiquitous nature of Internet services across the globe has undoubtedly expanded the strategies and operational mode being used by cybercriminals to perpetrate their unlawful activities through intrusion on various networks. Network intrusion has led to many global financial loses and privacy problems for Internet users across the globe. In order to safeguard the network and to prevent Internet users from being the regular victims of cyber-criminal activities, new solutions are needed. This research proposes solution for intrusion detection by using the improved hashing-based Apriori algorithm implemented on Hadoop MapReduce framework; capable of using association rules in mining algorithm for identifying and detecting network intrusions. We used the KDD dataset to evaluate the effectiveness and reliability of the solution. Our results obtained show that this approach provides a reliable and effective means of detecting network intrusion.

Keywords: association rule mining; intrusion detection; cyberattack; network security; apriori; MapReduce

1. Introduction

The survival of any business organization depends upon the security mechanisms that adequately protect and prevent from illegal entrance into confidential data of the organization. However, it might appear impossible to entirely control the breaches in security at present. On the back of this, researchers can attempt to detect intrusions and act accordingly to counter actions that brought about them. Intrusion detection scrutinizes computer activities for the purpose of uncovering violations [1]. The activity is especially relevant for new technologies such as smartphones [2], cloud computing [3], fog computing [4], and edge computing [5], where private and business data is shared across computer or wireless sensor networks, thus increasing the likelihood of attacks [6].

The intrusion detection system (IDS) provides insight into tracking and analysis of system and user activity, looking out for vulnerability, statistical anomalies, and performing behavioral analysis of user activities. The IDS can add expertise to the remaining part of critical network infrastructure and can follow activities of users from entry to action points. Being active, they can report alterations to administrators; when the system is under attack or when it detects errors in system configuration, guide the administrator in establishing necessarily important policies best for the organization's safety and even give permission to non-expert staff to carry out security management on the system. Even

though they may not be the all-in-all solution to all penetration threats and vulnerabilities, regular IDS can serve the purpose of ensuring that our system is secure to an appreciable level [7]. While the IDS are a sophisticated tool for enhancing protection, they do not compensate for weak mechanisms in user identification and authentication, in network protocols used and in integrity of information provided by the system. Further to this, they cannot investigate into issues of penetration without involvement of a network administrator; cannot examine all traffic on busy networks; cannot perpetually handle problems having to do with packet-level data and certainly cannot work with some of the modern network hardware and features. Recently, the researchers have grown more interest in the topics of detection of an intrusion in order to safeguard networks and their components [8–10]. IDS can be grouped into network-based (responsible for checking network structures and look out for attack signatures), host-based (function on host systems and function to defend systems from traces of intrusion), or application-based (aimed at monitoring specific applications and programs).

The signature-based IDS [11] use prior knowledge from a database of former attack signatures and recognized threats. The term signature implies a pool, where a set of stored proof of an attack are availed, since each passing leaves an image behind its attack. These left-back prints can be used in identifying and preventing same attacks later. The major demerit of the signature-based IDS is that regular updating and maintenance is required, so unique attacks can be recognized.

The anomaly-based IDS [12] refer to a baseline or learned model of system's normal activity, a significant deviation from this baseline is used to recognize the active intrusion attempts and trigger the alarm. These IDS are usually associated with higher rate of false alarms.

The network IDS [13] (NIDS) are made up of a network sensor, a network interface card (NIC) and an interface for management. The NIDS is fitted within a network segment and observes all traffic within that segment. NIDS recognizes intrusions by tracking network traffic and monitoring many hosts. With NIDS, sensors are put at stifle locations in the network to be checked, e.g., in the demilitarized zone (DMZ) or at network borders. The sensors function to capture all network traffic and inspect network packets for anything suspicious. NIDS can supervise an entire, large network using a small number of well-placed nodes or devices and subject the network to a little overhead. They act passively by not adding notable overhead or disrupting network processes, but just monitor network activities. The ease to secure a NIDS makes them mostly unrecognizable to potential attackers and they are easy to deploy and use on computer networks. While NIDS function extensively in supervising a large network, they are not able to track and analyze all network traffic and may not take note of attacks carried out at busiest periods and may not be able to safeguard high-load networks adequately. NIDS cannot analyze encrypted data neither do they report the outcome of attacks. Therefore, they may require active human involvement to minimize the effects of identified attacks.

The host intrusion detection systems (HIDS) [14] and several applications (agents) run on workstations to be mounted is the focal point. They mainly consist of an agent on a host system that recognizes system calls, application logs, system modifications, and other activities engaged in on the host system. The agents take note of the operating system and transfer data to log files and/or trigger alarms (if it is either active or passive IDS). The HIDS can only monitor on individual workstations, which have the agents running on them; it cannot function for the entire network. Consequently, they are used in capturing any intrusion attempts on crucial servers. However, the HIDS suffer a few demerits: difficulty in analyzing intrusion attempts on many systems, the presence of different operating systems might pose a challenge in maintenance in large networks, as the HIDS can be disabled by attackers on compromised systems.

As network topology and network traffic can change over time, the ability of these intrusion detection techniques to accommodate these changes decreases. Therefore, developing the IDS that can learn all network traffic patterns is nearly impossible. Several forms of the IDS have been brought forward to lighten the issues that they endure like accuracy, large volumes of data, distribution of these datasets, unease in recognizing the borders between the normal and abnormal behaviors in data sequences and adjusting to the regular changes in the environment [15].

Lately, new threats related with cloud computing, fog computing, and edge computing have emerged as these computing technologies are still vulnerable to security deficiencies and vulnerabilities for intrusion and other malicious activities affecting the integrity and availability of resources on cloud. For example, Abadeh and Habibi [16] proposed using evolutionary fuzzy rules and optimized GA for intrusion detection. Al Haddad et al. [17] introduced a collaborative network intrusion detection system (C-NIDS) to find network attacks, while addressing intrusion detection in virtual network, and other security challenges. Das et al. [18] used principal component analysis (PCA) to lower the complexity of the network data and detect network intrusions. Huang et al. [19] demonstrated the use of rough sets and support vector machines (SVM) for intrusion detection, while Khamphakdee et al. [20] used the association rules for identifying probe attacks. Kola Sujatha et al. [21] used a combination of SVM, fuzzy logic and genetic network programming (GNP) to create rules to detect the network intrusions. Hashem et al. [22] used the Bee Ranker (BR) algorithm based on the foraging behavior of honeybees for selection of features useful for detection of network intrusions. Gao et al. [23] combined an adaptive principal component (A-PCA) for adaptive selection of network traffic features, and incremental extreme learning machine (I-ELM) for intrusion detection. Abdulhammed [24] used auto-encoder (AE) and principle component analysis (PCA) to lower feature dimensionality before building Bayesian network, random forest (RF), linear discriminant analysis (LDA), and quadratic discriminant analysis (QDA) classifiers for an IDS. Al Tobi and Duncan [25] explored threshold adaptation for C5.0, random forest and SVM in improving the accuracy of the network intrusion detection models.

The Apriori algorithm was used in the context of network security before. Prasenna et al. [26] used fuzzy Weighted Association rule (WAR) based on the Apriori algorithm and genetic network programming (GNP) and for generation of rules to evaluate network misuse and anomaly detection. Lalli et al. [27] used GA to reduce network traffic features, then Apriori Association is used to generate a rule schema, and artificial neural network (ANN) filters the rules to increase the detection accuracy. An et al. [4] proposed using a hypergraph clustering model based on the Apriori algorithm to find the association between fog computing nodes with a higher risk of threat of the distributed denial of service (DDoS) attack. Jie et al. [28] proposed a similarity factor for quantifying the similarity between current and past frequent itemsets in real time. The factor is used as a reliability parameter to identify the abnormal system state.

For network IDSs, different frequent itemset mining algorithms, including the modifications of the Apriori algorithm, have been widely used. Yan et al. [29] employed the PrefixSpan-based sequence mining algorithm for network log analysis and intrusion discovery. Ohrui et al. [30] combined Apriori with Prefixspan, the sequential pattern mining approach, to detect and predict future botnet attacks. Zeng et al. [31] used sparse matrices to optimize Apriori for big data computing. The improved algorithm can save the runtime of data mining in intrusion detection, and save storage space for the analyzed network data. Khalili et al. [32] used Apriori to reduce the number of candidate states of industrial systems evaluated as critical. Zheng et al. [33] utilized the relational algebra theory to obtain the optimization relation association rule (ORAR) based on the relationship matrix and correlation operation, which reduces the number of dataset scanning operations to one, and thus overcomes the disadvantage of Apriori, which requires multiple scans.

Artificial intelligence (AI), machine learning (ML), and ANNs have become key research topics in the domain of anomaly-based intrusion detection. Chiba et al. [34] used the signature-based detection to find known attacks, and back-propagation neural network (BPN) classifier to identify unknown attacks. Odusami et al. [35] employed long short term memory (LSTM) for protecting against layer seven distributed denial of service (L7DDoS) intrusion attack. Yang et al. [36] employed modified density peak clustering algorithm (MDPCA) to lower the size of the training set and to deal with the class imbalance problem. Each cluster was used to train its own deep belief network (DBN) classifier. Finally, the decisions of the DBN classifiers is combined based on fuzzy membership weights, which are calculated using the nearest neighbor criterion. Le et al. [37] used the hybrid sequence forward selection and decision tree (SFSDT) model to generate the optimal feature subset for training recurrent

neural network (RNN) models. The latter are used to deal with user-to-root (U2R) and remote-to-local (R2L) attacks.

Summarizing, despite the plethora of methods used, including the advanced deep learning models, which have been proven successful in multiple other domains of application, they often fail for the real-world network intrusion detection task despite showing excellent results on benchmark network intrusion datasets. The reason is that the need of adaptation to pattern variability has often been neglected. Certain classes of attack such as denial of service (DoS) attacks are formed of abrupt patterns, which bring a high level of variability into network traffic patterns. For machine learning models, adaptation is performed only once using the validation data for cross-validation such as k-folds. However, in the real-world scenarios, validation data changes continuously, which makes such approaches inefficient [26].

Our novelty and contribution are the proposed improved hashing-based Apriori algorithm and its implementation on the MapReduce framework. The hashing-based modification allows too find the frequency of the k-itemsets without the use of computationally expensive candidate sets, which makes it usable for detecting network attacks in near real-time, whereas the MapReduce framework allows to handle network traffic on large networks. We demonstrate the applicability of the proposed method to identify and detect network intrusions using the KDD dataset as a benchmark.

2. Methodology

2.1. Apriori Algorithm

The Apriori algorithm is used for mining frequent itemsets based on the Boolean rules. The Apriori algorithm is considered as the most recognized algorithm to mine association rules. Developed by Agrawal and Sriknat [38], the algorithm finds association rules issues on a large scale, giving room to implicative outcomes that possess more than one element. Association rules seek frequent itemsets that have occurrences that go beyond a pre-defined least threshold and obtaining association rules from those frequency itemsets. These two sub-issues are solved repeatedly until no new rules appear. The least support threshold must be set by the user. The algorithm of Apriori is summarized in Algorithm 1. The algorithm has two stages: a training phase and a testing phase. In the training Stage, the algorithm can observe specs of behavior and the makes a generalization from it. Several algorithms organized learning stage, whereby samples of known attacks are supplied. In the testing stage, the algorithm is provided with a situation and it decides on the possibility of having an attack.

Algorithm 1: Apriori algorithm

Variables:
C_k is a candidate itemset of size k
L_k is a frequent itemset of size k
BEGIN
  Find frequent set L_{k-1}
  Generate C_k by using Cartesian product of L_{k-1}, i.e. L_{k-1} x L_{k-1}
  Perform pruning: remove any k−1 size itemsets that are not frequent
  Return frequent set L_k
END

The algorithm utilizes a breadth-first search mechanism and a hash tree configuration to make candidate itemsets counted efficiently to determine the frequency of occurrence for each itemsets. The pseudocode of the algorithm is summarized in Algorithm 2.

Algorithm 2: Find frequent itemsets

BEGIN
$L_1 \leftarrow \{large\ 1-itemsets\ that\ appear\ in\ more\ than\ \varepsilon\ transactions\}$
$k \leftarrow 2$
while $L_{k-1} \neq \oslash$
$C_k \leftarrow Generate(L_{k-1})$
for transactions $t \in T$
$C_t \leftarrow Subset(C_k, t)$
for candidates $c \in C_t$
$count[c] \leftarrow count[c] + 1$
$L_k \leftarrow \{c \in C_k \mid count[c] \geq \varepsilon\}$
$k \leftarrow k + 1$
return $\bigcup_k L_k$
END

The disadvantage of the classical Apriori algorithm is rapid performance degradation when working with very large datasets because of recurring scanning of the dataset and the creation of many candidate sets. To improve the efficiency of the Apriori algorithm, we have adopted the idea of hashing first presented by Tribhuvan et al. [39]. The Modified Apriori algorithm employs hash tables to generate large itemsets efficiently. It runs over the entire dataset and stores previous results in the hash tables. This allows to void repeating scan as the results stored in hash tables are used. We also employ the double hashing techniques by Jayalakshmi et al. [40]. First, the data is represented using the Transaction id format. The hashing is used to store the data values. To resolve hashing collisions, an independent second hashing function is used. Following the suggestion of [41], for hashing we adopt Hamming projections, which are described as follows:

$$\mathcal{H}_A\{x \rightarrow x \wedge a | a \in A\},\ A \subseteq \{0,1\}^{(1+Smax)n}, \tag{1}$$

here *Smax* is maximum support, and n is the count of transactions.

The hash table construction procedure has two parts: generation of hash value and update of the hash table. Similarly to Reference [42], to increase the speed of hash table construction, we employ parallel hash value generators to allow for the simultaneous generation of hash values. However, instead of hardware parallelization, we use MapReduce [43]. The MapReduce model is based on the division of the large dataset into smaller data subset. Then the Map function is used to parallelize the processing of each data subset. The Reduce function performs the combining of the results. In case of the Apriori algorithm, we follow the suggestion of Zhou et al. [44] and start from the frequent 2-itemset. As a key value, we use the first (k − 2) term of the frequent (k − 1)-itemset, whereas the value is the last term. The Reduce function combines the results into previous (k − 2) items. Thus, our method has two stages. First, MapReduce is applied to calculate frequent itemsets in parallel. Then the frequent itemsets are subjected to MapReduce again to find the association rules. The pruning is performed based on support and confidence threshold criteria. The algorithm is summarized as a flow diagram in Figure 1.

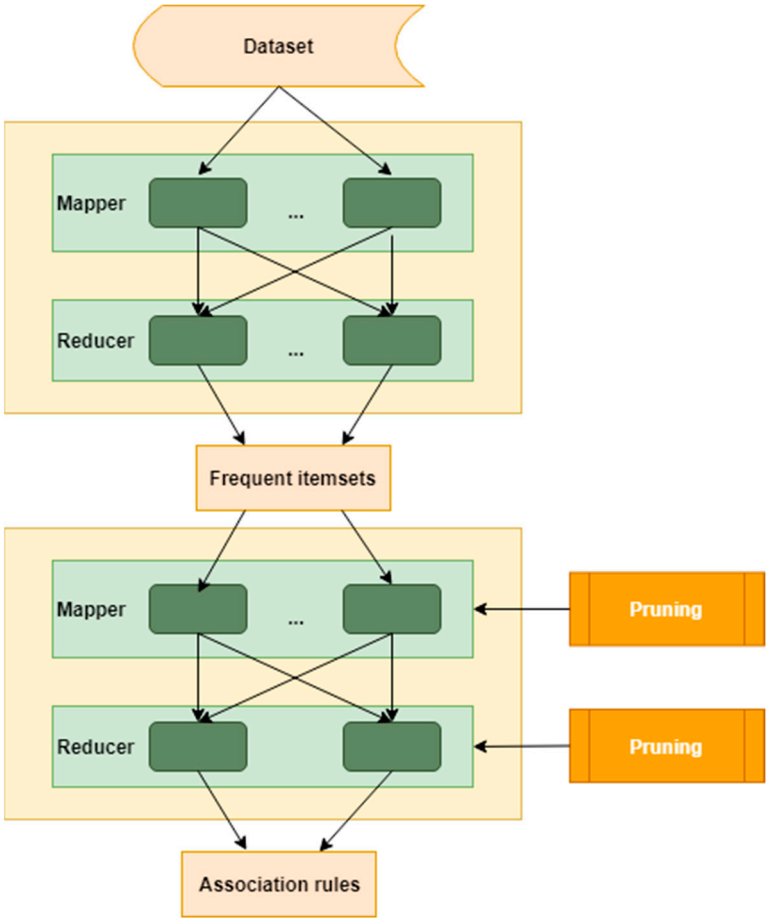

Figure 1. Flow diagram of the improved Apriori algorithm.

2.2. Dataset

The KDDcup99 intrusion dataset (available from the UCI KDD repository) was used for the testing of the algorithm. The dataset has many intrusion attacks simulated in a military-grade network. The KDDcup99 dataset is considered as the benchmark tagged dataset [45] that is commonly used for assessing the network intrusion detection methods. Here we use the 10% KDD training data subset. The dataset has some important features, which support the purpose of its usage for intrusion detection. TCP packets are used to describe the connections from the initial stage to the end at important acceptable intervals. Each connection is tagged and labeled as normal or abnormal. There are 494,021 connection attempts at the LAN. The dataset covers four main attack types (see Table 1) and has 41 features separated into both continuous and discrete sets. The 41 features (see Table 2) include the characteristics of TCP connections, content and network traffic features, which are calculated with a 2 sec time window [46].

Table 1. Attack types in the KDDcup99 intrusion dataset.

Class	Description	Number of Records	Attack Types
DOS	denial of services (DoS),	391,458	Teardrop, smurf, pod, Neptune, Land, Back
U2R	unauthorized access to local supervisor privileges	52	Rootkit, perl, loadmodule, Buffer_overflow,
R2L	unauthorized access from a remote machine	1126	Warezmaster, warezlient, spy, phf, multihop, imap, guess_passwd, ftp_write
Probe	surveillance and other probing	4107	satan, portsweep, nmap, IPsweep

Table 2. Features used for constructing Apriori rules.

Class	Selected Features
DoS	count, dst_bytes, dst_host_count, dst_host_serror_rate, dst_host_srv_count, dst_host_srv_serror_rate, flag, protocol type, serror_rate, service, src_bytes, srv_count, srv_serror_rate.
Probe	count, diff_srv_rate, dst_bytes, dst_host_count, dst_host_diff_srv_rate, dst_host_rerror_rate, dst_host_same_src_port_rate, dst_host_same_srv_rate, dst_host_srv_count, dst_host_srv_diff_host_rate, dst_host_srv_rerror_rate, dst_host_srv_serror_rate, duration, flag, protocol_type, rerror_rate, same_srv_rate, service, src_bytes, srv_count, srv_diff_host_rate, srv_rerror_rate, srv_serror_rate.
R2L	count, dst_host_count, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_same_srv_rate, dst_host_srv_count, dst_host_srv_diff_host_rate, flag, hot, is_guest_login, logged_in, same_srv_rate, services.
U2R	count, dst_bytes, dst_host_count, dst_host_same_src_port_rate, dst_host_same_srv_rate, dst_host_srv_count, duration, flag, hot, logged_in, num_compromised, num_file_creations, num_root, num_shells, protocol_type, root_shell, same_srv_rate, service, src_bytes, srv_count.

2.3. Evaluation

We evaluate the results using the strength measures of association rules, i.e., support and confidence. Support defines how often a rule can be applied to a dataset, whereas confidence defines how often items in consequent of the rules appear in the rules that contain the antecedent:

$$s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}, \tag{2}$$

$$c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}, \tag{3}$$

here S is the support, C is the confidence, X is the antecedent, Y is the consequent, and σ is the frequency of the itemset.

A rule that has a low level of support may be occurring by chance. Confidence measures reliability of the rule and provides an estimate of the conditional probability of consequent upon the antecedent.

3. Results

We adopted Tanagra 1.4.50 (Lumière University Lyon 2, Lyon, France) on a Windows 10 OS, Inter Core i7 2.7 GHZ, 16GB RAM as the main software platform for the implementation of this research. The tool is used for evaluating both multivariate and univariate parametric and nonparametric tests, and for the extraction of results, in the form of rules, for the Apriori algorithm. We employed the tool to perform operations on the KDDcup99 dataset to get results from the various network cyberattack attempts.

Validation has been performed using 10-fold cross-validation. The technique allows to obtain the accuracy results that are less sensitive with regard to different training subsets. In 10-fold cross validation, traffic profiles are split into ten sets and a training set is made by joining nine randomly selected sets. The remaining subset is utilized as testing set for assessing the classification performance. The entire process is replicated ten times by joining the subsets in ten different ways, and the mean accuracy rate is computed.

The results (the rules with largest confidence) are presented in Table 3 below with the following parameters of the Apriori algorithm (Max rule length 4, Support min 0.33, Lift filtering 1.1, Confidence min 0.75). The algorithm generates 146 rules (based on 494,020 transactions).

Figure 2 explains relationship between the parameters and how they served to detect strange signatures in the database for attack types given in Table 1. Note that R2L attacks have been identified with a higher level of support than other types of attack (DOS, U2R, Probe, see Table 1), while DOS attacks had the lowest level of support.

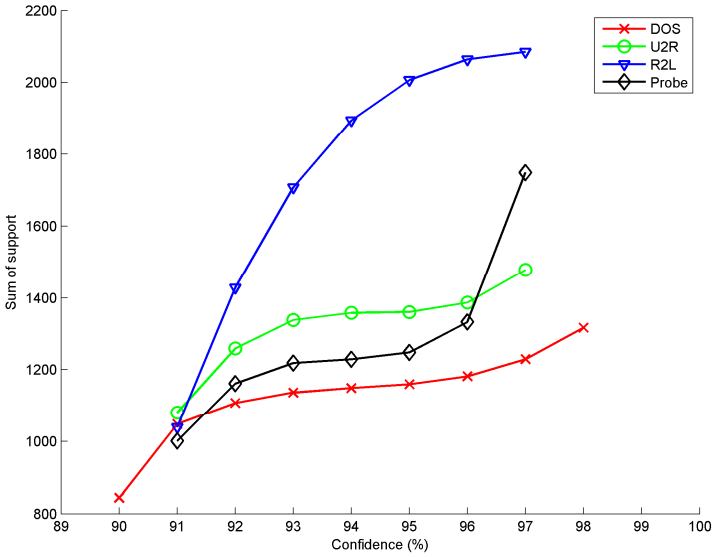

Figure 2. Confidence vs. sum of support for different types of network attack.

For different attack types, the detection rate is provided as radar diagram in Figure 3.

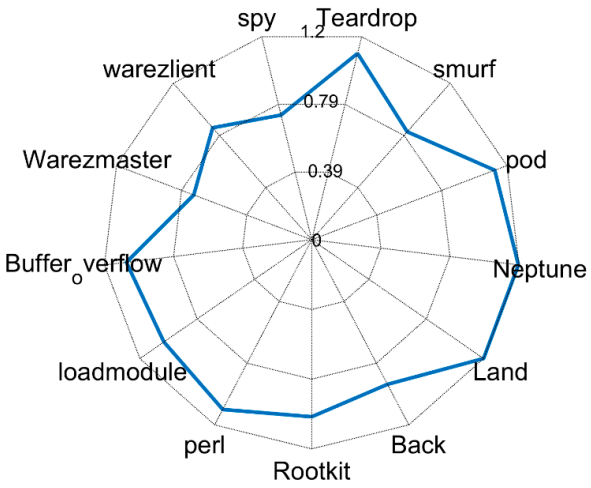

Figure 3. Accuracy of attack type detection.

Table 3. Sample of Apriori Results

Antecedent	Consequent	Support (%)	Confidence (%)
"same_srv_rate = true" - "dst_host_same_src_port_rate = true"	"src_bytes = true"	69.646	97.988
"srv_count = true" - "same_srv_rate = true" - "dst_host_same_src_port_rate = true"	"src_bytes = true"	69.646	97.988
"same_srv_rate = true" - "src_bytes = true"	"srv_count = true" - "dst_host_same_src_port_rate = true"	69.646	90.860
"count = true" - "dst_host_same_src_port_rate = true"	"same_srv_rate = true" - "src_bytes = true"	69.646	97.979
"same_srv_rate = true" - "src_bytes = true"	"count = true" - "dst_host_same_src_port_rate = true"	69.646	90.860
"srv_count = true" - "dst_host_same_src_port_rate = true"	"same_srv_rate = true" - "src_bytes = true"	69.646	97.979
"dst_host_same_src_port_rate = true"	"dst_host_count = true" - "same_srv_rate = true" - "src_bytes = true"	69.646	97.979
"dst_host_srv_count = true" - "same_srv_rate = true" - "src_bytes = true"	"dst_host_same_src_port_rate = true"	69.646	90.861
"dst_host_count=true" - "same_srv_rate = true" - "src_bytes = true"	"dst_host_same_src_port_rate = true"	69.646	90.861
"dst_host_same_src_port_rate = true"	"dst_host_srv_count = true" - "same_srv_rate = true" - "src_bytes = true"	69.646	97.979
"same_srv_rate = true" - "src_bytes = true"	"dst_host_count = true" - "dst_host_same_src_port_rate = true"	69.646	90.86
"dst_host_count = true" - "dst_host_same_src_port_rate = true"	"same_srv_rate = true" - "src_bytes = true"	69.646	97.979

4. Evaluation

4.1. Accuracy

We evaluate our results against state-of-the-art of other authors achieved using a variety of different methods on KDDcup99 dataset (see Table 4). Note that the perfect result achieved by some of the other methods does not mean that the corresponding method will behave well in a real-world situation, where network traffic patterns constantly change.

Table 4. Comparison of intrusion detection rate.

Authors	Method	DOS	U2R	R2L	PRB
Hadri et al. [47]	Robust fuzzy PCA	74.2	16.1	4.55	92.1
Papamartzivanos et al. [48]	Genetic trees	99.12	52.63	79.54	82.63
Elhag et al. [49]	Evolutionary fuzzy	90.28	42.31	92.77	67.15
Le et al. [37]	Recurrent Neural Networks (RNN)	88	85	81	100
Zhao et al. [45]	SVM+RBF kernel	99.77	96.19	91.07	n/a
This method	Apriori	98.2	98.1	96.9	96.9

Finally, we present the confusion matrices for DOS, U2R, R2L, and PRB attack classification in Figure 4. The accuracy for recognition of different types of attack is similar, with the DOS attacks recognized with a highest accuracy of 98.2%, and the PRB attacks recognized at a lowest accuracy of 96.91%.

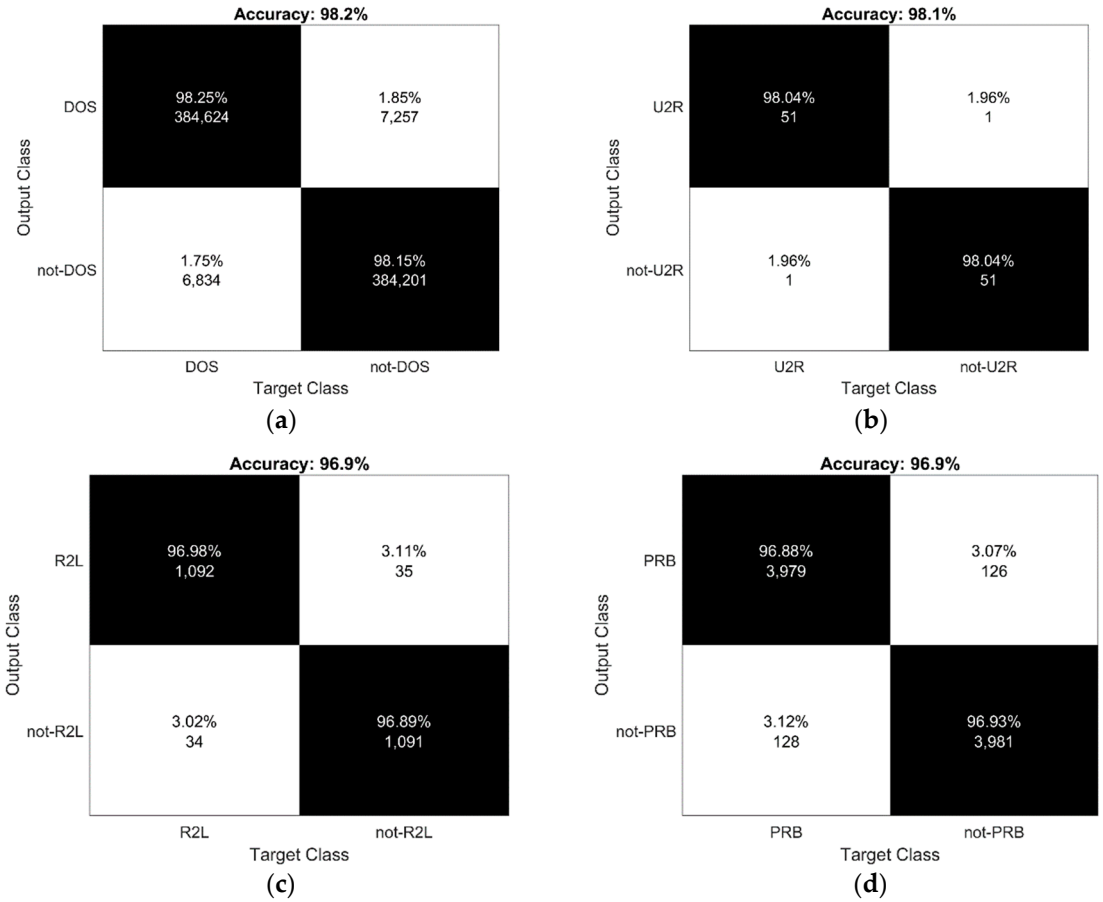

Figure 4. Confusion matrices for (**a**) DOS, (**b**) U2R, (**c**) R2L, and (**d**) PRB attack classification.

Following the recommendation of Demšar [50], we used a series of statistical tests to compare the methods. The Friedman Test ranks the algorithms by assigning a rank for performance of each method

for each dataset. The Nemenyi post-hoc test was applied to compute an average ranking difference threshold as critical distance (CD). The hypothesis that "the accuracy of two methods is the same" is rejected, if their mean rank difference is larger than CD. The results are summarized as the Demšar significance diagram in Figure 5. Considering different types of attack, on average, we proposed method performs better than other considered methods. However, statistically the ranking of the proposed method is statistically undistinguishable from the methods of Zhao et al. [45], Le et al. [37], and Papamartzivanos et al. [48].

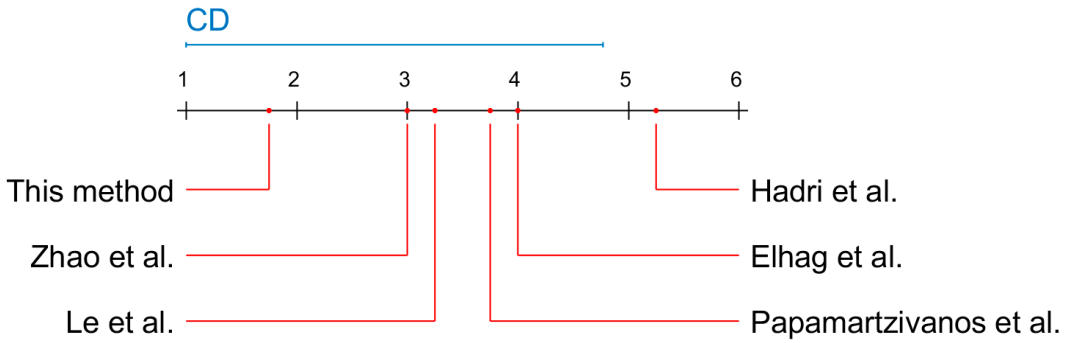

Figure 5. Demšar significance diagram for comparison of methods using Nemenyi test.

4.2. Scalability

To analyze and evaluate the performance and scalability of the proposed solution, we have set-up a cluster of eight PC nodes. Each node runs Microsoft Windows 10 Home operating system on Intel i5-8265U CPU, 1.60GHz (4 cores, 8 logical processors) with 8GB RAM and 15.6 GB virtual memory available. All algorithms were implemented using Python version 3.7.4. For implementation of MapReduce, we used Apache Hadoop 1.2.1 framework. To test the solution, we used the full KDDcup99 dataset, which has about five mln. records in the training part, and around two mln. records in the testing part. To evaluate scalability, we used the Speedup measure [51], which is the ratio of performance on a single-node system with respect to performance on an n-node system. Speedup is measured by evaluating the performance of the framework on the dataset by the number of nodes.

In Figure 6, we report the running time results for a different number of computing nodes. The impact of using MapReduce on the running time can be seen. The running time on 2 nodes takes 1875 s, while the running time on 8 nodes takes 420 s for the same dataset. The speedup factor of the running times for 4 nodes as compared to the runtime with a single node is 3.78, and for 8 nodes, the improvement is 7.59. As we can observe from Figure 6, the speedup is very close to the linear one when using from 2 to 8 nodes. The results demonstrate reasonable scalability for the suggested network intrusion detection system.

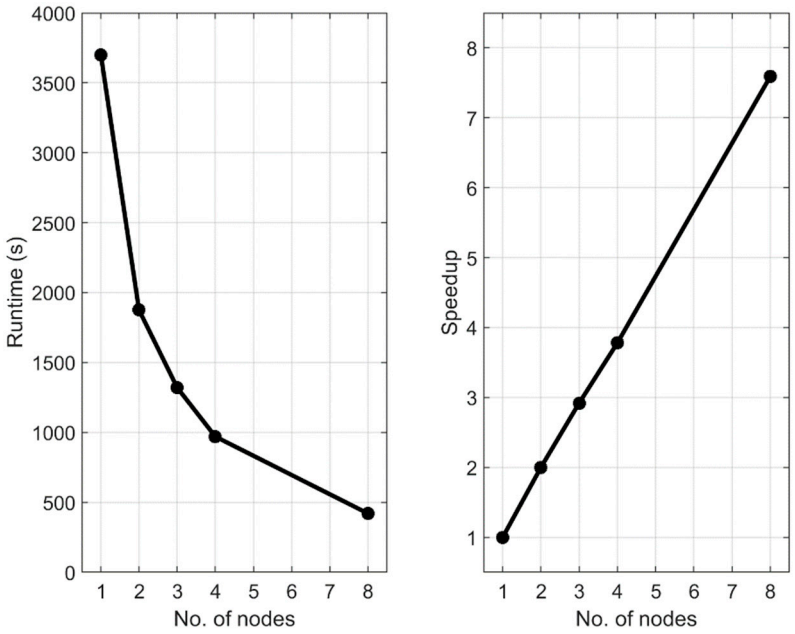

Figure 6. Runtime and speedup results on the full KDDCup99 dataset.

5. Discussion and Conclusions

One of the main shortcomings of the classical Apriori algorithm is inefficient performance when working with big datasets, because of repeated scanning of the database and the creation of many candidate sets. In this paper we tackled this problem by adopting a hashing approach, which allows to find the frequency of the k-itemsets without the use of computationally expensive candidate sets. The hashing-based approach also has an advantage in its high-computation speed, which has already been noted by other researchers [52–54]. The latter makes it usable for detecting network attacks in near real-time, while the adoption of MapReduce allows for scalability with big data [55,56].

We have described the intrusion detection framework for identification of network intrusions. The framework uses the Apriori algorithm for intrusion detection, to find attacks and develop rules, and has an appreciable level of accuracy and efficiency in finding out new cyberattacks using the pieces of information provided about known and recognized attacks. The framework was applied on the KDDcup99 dataset and provided successful recognition of four types of network attacks with high confidence and level of support.

The proposed method can produce solutions that address the shortcomings of other approaches, specifically, the lack of adaptability demonstrated by the neural network based methods. The proposed methodology based on double hashing is promising and can be used for detecting cyber-attacks. However, the association rules do not imply causality, but rather the co-occurrence of events. Moreover, the researchers need better and more standard datasets that are presently prevalent and indicative of today's web servers. Researchers and business organizations need to look through network defense mechanisms with a view to identifying loopholes and improving the system to provide a more reliable protection from cyber-attacks.

In future works, we will perform a more in-depth research on the recognition of cyber-attacks in edge and fog computing environments.

Author Contributions: Conceptualization, N.A.A. and S.M.; methodology, S.M.; software, N.A.A. and T.J.A.; validation, N.A.A., S.M., R.M., and R.D.; formal analysis, N.A.A. and S.M.; investigation, N.A.A., T.J.A., S.M., and R.D.; resources, N.A.A., and S.M.; writing—original draft preparation, N.A.A. and S.M.; writing—review and editing, R.M. and R.D.; visualization, R.D.; supervision, S.M.

Funding: This research was funded by H2020 LEIT Information and Communication Technologies, grant number 830892.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Amor, N.; Benferhat, S.; Elouedi, Z. Naïve Bayes vs. decision trees in intrusion detection systems. In Proceedings of the 2004 ACM Symposium on Applied Computing, New York, NY, USA, 14–17 March 2004; pp. 420–424.
2. Odusami, M.; Abayomi-Alli, O.; Misra, S.; Shobayo, O.; Damasevicius, R.; Maskeliunas, R. Android malware detection: A survey. In *Communications in Computer and Information Science*; Springer International Publishing: Cham, Switzerland, 2018; pp. 255–266. [CrossRef]
3. Odun-Ayo, I.; Geteloma, V.; Misra, S.; Ahuja, R.; Damasevicius, R. Systematic Mapping Study of Utility-Driven Platforms for Clouds. In *Proceedings of ICETIT 2019*; Springer International Publishing: Cham, Switzerland, 2019; pp. 762–774. [CrossRef]
4. An, X.; Su, J.; Lü, X.; Lin, F. Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system. *Eurasip J. Wirel. Commun. Netw.* **2018**, 1. [CrossRef]
5. Venčkauskas, A.; Morkevicius, N.; Jukavičius, V.; Damaševičius, R.; Toldinas, J.; Grigaliūnas, Š. An Edge-Fog Secure Self-Authenticable Data Transfer Protocol. *Sensors* **2019**, *19*, 3612. [CrossRef] [PubMed]
6. Wei, W.; Woźniak, M.; Damaševičius, R.; Fan, X.; Li, Y. Algorithm research of known-plaintext attack on double random phase mask based on WSNs. *J. Internet Technol.* **2019**, *201*, 39–48. [CrossRef]
7. Bai, Y.; Kobayashi, H. Intrusion detection system: Technology and developments. In Proceedings of the 17th International Conference on Advanced Information Networking and Application, 2003. AINA 2003, Xi'an, China, 29 March 2003; p. 710.
8. Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network intrusion detection for IoT security based on learning techniques. *IEEE Commun. Surv. Tutor.* **2019**, *213*, 2671–2701. [CrossRef]
9. da Costa, K.A.P.; Papa, J.P.; Lisboa, C.O.; Munoz, R.; de Albuquerque, V.H.C. Internet of things: A survey on machine learning-based intrusion detection approaches. *Comput. Netw.* **2019**, *151*, 147–157. [CrossRef]
10. Kwon, D.; Kim, H.; Kim, J.; Suh, S.C.; Kim, I.; Kim, K.J. A survey of deep learning-based network anomaly detection. *Clust. Comput.* **2019**, *22*, 949–961. [CrossRef]
11. Uddin, M.; Rehman, A.A.; Uddin, N.; Memon, J.; Alsaqour, R.; Kazi, S. Signature-based multi-layer distributed intrusion detection system using mobile agents. *Int. J. Netw. Secur.* **2013**, *15*, 97–105.
12. Patcha, A.; Park, J. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.* **2007**, *51*, 3448–3470. [CrossRef]
13. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network anomaly detection: Methods, systems and tools. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 303–336. [CrossRef]
14. Liu, M.; Xue, Z.; Xu, X.; Zhong, C.; Chen, J. Host-based intrusion detection system with system calls: Review and future trends. *ACM Comput. Surv.* **2019**, *51*. [CrossRef]
15. Debar, H.; Dacier, M.; Wespi, A. *A Revised Taxonomy for Intrusion Detection Systems*; Springer International Publishing: Cham, Switzerland, 2000; pp. 361–378.
16. Abadeh, M.; Habibi, J. *A Hybridization of Evolutionary Fuzzy Systems and Ant Colony Optimization for Intrusion Detection*; Sharif University of Technology: Tehran, Iran, 2010.
17. Al Haddad, Z.; Hanoune, M.; Mamouni, A. A collaborative network intrusion detection system (C-NIDS) in cloud computing. *Int. J. Commun. Netw. Inf. Secur.* **2016**, *8*, 130–135.
18. Das, A.; Nguyen, D.; Zambreno, J.; Memik, G.; Choudhary, A. An FPGA-based network intrusion detection architecture. *IEEE Trans. Inf. Forensics Secur.* **2008**, *3*, 118–132. [CrossRef]
19. Huang, J.; Chen, C. Integration of rough sets and support vector machines for network intrusion detection. *J. Ind. Prod. Eng.* **2014**, *31*, 425–432. [CrossRef]
20. Khamphakdee, N.; Benjamas, N.; Saiyod, S. Improving intrusion detection system based on snort rules for network probe attacks detection with association rules technique of data mining. *J. ICT Res. Appl.* **2015**, *8*, 234–250. [CrossRef]
21. Kola Sujatha, P.; Suba Priya, C.; Kannan, A. Network intrusion detection system using genetic network programming with support vector machine. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ACM International Conference Proceeding Series, New York, NY, USA, 3–5 August 2012; pp. 645–649. [CrossRef]

22. Hashem, S.H. Enhance network intrusion detection system by exploiting br algorithm as an optimal feature selection. In *Handbook of Research on Threat Detection and Countermeasures in Network Security*; Information Science Reference: Hershey, PA, USA, 2014; pp. 17–32. [CrossRef]
23. Gao, J.; Chai, S.; Zhang, B.; Xia, Y. Research on Network Intrusion Detection Based on Incremental Extreme Learning Machine and Adaptive Principal Component Analysis. *Energies* **2019**, *12*, 1223. [CrossRef]
24. Abdulhammed, R.; Musafer, H.; Alessa, A.; Faezipour, M.; Abuzneid, A. Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection. *Electronics* **2019**, *8*, 322. [CrossRef]
25. Al Tobi, A.M.; Duncan, I. Improving Intrusion Detection Model Prediction by Threshold Adaptation. *Information* **2019**, *10*, 159. [CrossRef]
26. Prasenna, P.; Kumar, R.K.; Ramana, A.V.T.; Devanbu, A. Network programming and mining classifier for intrusion detection using probability classification. In Proceedings of the International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012), Salem, Tamilnadu, India, 21–23 March 2012; pp. 204–209.
27. Lalli, M.; Palanisamy, V. Filtering framework for intrusion detection rule schema in mobile ad hoc networks. *Int. J. Control Theory Appl.* **2016**, *9*, 195–201.
28. Jie, X.; Wang, H.; Fei, M.; Du, D.; Sun, Q.; Yang, T.C. Anomaly behavior detection and reliability assessment of control systems based on association rules. *Int. J. Crit. Infrastruct. Prot.* **2018**, *22*, 90–99. [CrossRef]
29. Yan, S.; Chen, Y.; Song, Y.; Zhu, M. Frequent attack sequences-based network log mining. *J. Phys. Conf. Ser.* **2019**, *1176*. [CrossRef]
30. Ohrui, M.; Kikuchi, H.; Rosyid, N.R.; Terada, M. Mining botnet coordinated attacks using apriori-prefixspan hybrid algorithm. *J. Inf. Process.* **2013**, *21*, 607–616. [CrossRef]
31. Zeng, X.; Lv, J.; Li, J.; Luo, W. An optimized apriori algorithm based on sparse matrix for intrusion detection. *Open Cybern. Syst. J.* **2014**, *8*, 8–11.
32. Khalili, A.; Sami, A. SysDetect: A systematic approach to critical state determination for industrial intrusion detection systems using apriori algorithm. *J. Process Control* **2015**, *32*, 154–160. [CrossRef]
33. Zheng, J.; Yang, L. Research on the improvement of apriori algorithm and its application in intrusion detection system. In Proceedings of the 2015 IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 10–11 October 2015; pp. 105–108. [CrossRef]
34. Chiba, Z.; Abghour, N.; Moussaid, K.; El Omri, A.; Rida, M. A cooperative and hybrid network intrusion detection framework in cloud computing based on snort and optimized back propagation neural network. *Procedia Comput. Sci.* **2016**, *83*, 1200–1206. [CrossRef]
35. Odusami, M.; Misra, S.; Adetiba, E.; Abayomi-Alli, O.; Damasevicius, R.; Ahuja, R. An improved model for alleviating layer seven distributed denial of service intrusion on webserver. *J. Phys. Conf. Ser.* **2019**, *1235*. [CrossRef]
36. Yang, Y.; Zheng, K.; Wu, C.; Niu, X.; Yang, Y. Building an Effective Intrusion Detection System Using the Modified Density Peak Clustering Algorithm and Deep Belief Networks. *Appl. Sci.* **2019**, *9*, 238. [CrossRef]
37. Le, T.-T.-H.; Kim, Y.; Kim, H. Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks. *Appl. Sci.* **2019**, *9*, 1392. [CrossRef]
38. Agrawal, R.; Srikant, R. Fast algorithms for mining association rules in Large Databases. In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, San Francisco, CA, USA, 12–15 September 1994; pp. 487–499.
39. Tribhuvan, S.A.; Gavai, N.R.; Vasgi, B.P. Frequent Itemset Mining Using Improved Apriori Algorithm with MapReduce. In Proceedings of the 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 17–18 August 2017; pp. 1–6. [CrossRef]
40. Jayalakshmi, N.; Vidhya, V.; Krishnamurthy, M.; Kannan, A. Frequent Itemset Generation using Double Hashing Technique. *Procedia Eng.* **2012**, *38*, 1467–1478. [CrossRef]
41. Bera, D.; Pratap, R. Frequent-Itemset Mining Using Locality-Sensitive Hashing. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2016; pp. 143–155. [CrossRef]
42. Wen, Y.; Huang, J.; Chen, M. Hardware-enhanced association rule mining with hashing and pipelining. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 784–795. [CrossRef]
43. Dean, J. Experiences with MapReduce, an abstraction for large-scale computation. In Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques, Seattle, Washington, DC, USA, 16–20 September 2006; p. 1. [CrossRef]

44. Zhou, H.; Zhang, D.; Wang, X. Improvement of Apriori-Pro Algorithm Based on MapReduce. In *Advances in Intelligent Systems and Computing*; Springer International Publishing: Cham, Switzerland, 2019; pp. 1257–1265. [CrossRef]
45. Zhao, F.; Zhao, J.; Niu, X.; Luo, S.; Xin, Y. A Filter Feature Selection Algorithm Based on Mutual Information for Intrusion Detection. *Appl. Sci.* **2018**, *8*, 1535. [CrossRef]
46. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
47. Hadri, A.; Chougdali, K.; Touahni, R. Identifying intrusions in computer networks using robust fuzzy PCA. In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017; pp. 1261–1268.
48. Papamartzivanos, D.; Gómez Mármol, F.; Kambourakis, G. Dendron: Genetic trees driven rule induction for network intrusion detection systems. *Future Gener. Comput. Syst.* **2018**, *79*, 558–574. [CrossRef]
49. Elhag, S.; Fernández, A.; Altalhi, A.; Alshomrani, S.; Herrera, F. A multi-objective evolutionary fuzzy system to obtain a broad and accurate set of solutions in intrusion detection systems. *Soft Comput.* **2017**, 1–16. [CrossRef]
50. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
51. Aljarah, I.; Ludwig, S.A. MapReduce intrusion detection system based on a particle swarm optimization clustering algorithm. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 955–962. [CrossRef]
52. Rathinasabapathy, R.; Bhaskaran, R. Performance Comparison of Hashing Algorithm with Apriori. In Proceedings of the 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies (ACT 2009), Trivandrum, Kerala, India, 28–29 December 2009. [CrossRef]
53. Shakya, S.; Singh, A.; Singh, D. A Survey on Hash based A-priori Algorithm for Web Log Analysis. *Int. J. Comput. Appl.* **2013**, *76*, 47–50. [CrossRef]
54. Lin, C.-C.; Li, W.-C.; Chen, J.-C.; Chung, W.-Y.; Chung, S.-H.; Lin, K.W. A Distributed Algorithm for Fast Mining Frequent Patterns in Limited and Varying Network Bandwidth Environments. *Appl. Sci.* **2019**, *9*, 1859. [CrossRef]
55. Maitrey, S.; Jha, C.K. MapReduce: Simplified Data Analysis of Big Data. *Procedia Comput. Sci.* **2015**, *57*, 563–571. [CrossRef]
56. Veiga, J.; Exposito, R.R.; Pardo, X.C.; Taboada, G.L.; Tourifio, J. Performance evaluation of big data frameworks for large-scale data analytics. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 424–431. [CrossRef]