symmetry

MDPI

Article

Toward Bulk Synchronous Parallel-Based Machine Learning Techniques for Anomaly Detection in High-Speed Big Data Networks

Kamran Siddique [1], Zahid Akhtar [2], Haeng-gon Lee [3], Woongsup Kim [1] and Yangwoo Kim [1,*]

[1] Department of Information and Communication Engineering, Dongguk University, Seoul 04620, Korea; kamran@dongguk.edu (K.S.); woongsup@dongguk.edu (W.K.)
[2] INRS-EMT, University of Quebec, Montreal, QC H5A 1K6, Canada; zahid.akhtar.momin@emt.inrs.ca
[3] Korea Institute of Science and Technology Information, Daejeon 34141, Korea; hglee@kisti.re.kr
* Correspondence: ywkim@dongguk.edu; Tel.: +82-22-260-3821

Received: 28 August 2017; Accepted: 15 September 2017; Published: 19 September 2017

Abstract: Anomaly detection systems, also known as intrusion detection systems (IDSs), continuously monitor network traffic aiming to identify malicious actions. Extensive research has been conducted to build efficient IDSs emphasizing two essential characteristics. The first is concerned with finding optimal feature selection, while another deals with employing robust classification schemes. However, the advent of big data concepts in anomaly detection domain and the appearance of sophisticated network attacks in the modern era require some fundamental methodological revisions to develop IDSs. Therefore, we first identify two more significant characteristics in addition to the ones mentioned above. These refer to the need for employing specialized big data processing frameworks and utilizing appropriate datasets for validating system's performance, which is largely overlooked in existing studies. Afterwards, we set out to develop an anomaly detection system that comprehensively follows these four identified characteristics, i.e., the proposed system (i) performs feature ranking and selection using information gain and automated branch-and-bound algorithms respectively; (ii) employs logistic regression and extreme gradient boosting techniques for classification; (iii) introduces bulk synchronous parallel processing to cater computational requirements of high-speed big data networks; and; (iv) uses the Infromation Security Centre of Excellence, of the University of Brunswick real-time contemporary dataset for performance evaluation. We present experimental results that verify the efficacy of the proposed system.

Keywords: anomaly detection; network intrusion detection systems; bulk synchronous parallel; machine learning; big data; ISCX-UNB dataset; DARPA; KDD Cup 99

1. Introduction

This decade has witnessed tremendous growth in cyberspace and various computing devices. Proliferation of the Internet with these computing devices has enhanced efficiency and productivity in almost all the dimensions of life. Along with such advancements, mechanisms to cope with intrusive activities have become a primary concern for almost every individual and organizations in particular. An unintended access to organizational network can create havoc and even jeopardize activities. Therefore, regardless of the nature of the organization, detecting possible intrusions and protecting against external as well as internal attacks on networks is of paramount importance. In order to mitigate this problem, intrusion detection systems (IDSs), especially network intrusion detection systems (NIDSs) are widely implemented in various network environments. An IDS is a software or hardware component that aims at identifying malicious actions such as attempts to compromise the confidentiality, integrity or availability of a resource [1].

A significant amount of research has been conducted to develop efficient intrusion detection systems using various techniques such as statistical, soft computing, combination learners, and the methods based on classification, knowledge, clustering, and so on [2–6]. However, in the past few years, the exponential growth of massive data in network domain has posed many challenges to researchers in the field [7–9]. A large number of new data is being produced by high speed networks on daily basis. Security solutions such as NIDSs need to analyze huge traffic of such networks in real time, as the ever-increasing number of anomalies can have a destructive effect on confidentiality and availability of information. The situation has resulted in introducing the term 'big data' also in the field of intrusion detection. Current anomaly detection systems require frequent analysis of big data, which is an active research problem as it is very difficult or impossible for traditional technologies to handle such huge datasets [10,11]. Big data is generally represented by the mathematical relationship with three independent variables: volume, velocity, and variety of the datasets [12]. In anomaly detection domain, big data problem has its own issues. For instance, a large number of data can cause a chain reaction which reduces the efficiency of NIDSs. The main challenges and issues regarding network anomaly detection under big data environments are summarized in Table 1.

Table 1. Main issues regarding network anomaly detection under big data.

Dimension	Description	Challenges and Issues
Volume	The size of the datasets	• Can cause traffic overloads • Can make the processing capability slower
Velocity	The speed at which the data is being generated	• Difficult to handle traffic in real-time environment • May increase packet-drop ratio
Variety	The complexity of data	• Hard to perform feature selection operations • Many machine learning techniques may not be applicable
Veracity	Refers to the trustworthiness of the data in terms of accuracy	• May include data quality problems such as noise or missing values

During the past number of years, anomaly detection based on machine learning and data mining techniques have received considerable attention among researchers. More emphasis has been given on devising efficient feature selection schemes and robust classification methods, as they are generally considered most vital characteristics to build an efficient IDS. In recent years, the association of intrusion detection with big data domain has been recognized and researchers have started to deploy specialized big data frameworks attempting to handle computational requirements efficiently [13–15]. Despite great efforts, there are two important aspects that hinder the progress of NIDS research and greatly need the attention of IDS research community. They are concerned with the decision to select appropriate big data computing framework and to utilize adequate datasets for the evaluation of an IDS. We emphasize that the value and legitimacy of such decisions is equally important as other fundamental characteristics possess in the process of developing efficient IDSs. Building on the points addressed so far, we introduce a comprehensive IDS incorporating bulk synchronous parallel (BSP_based machine learning classification techniques and validate its effectiveness using an adequate modern dataset from the Information Security Centre of Excellence, of the University of Brunswick (ISCX, UNB) [16]. To the best of our knowledge, this article provides the first such mechanisms and settings for anomaly detection in big data networks. The key contributions of this paper are summarized as follows:

- We begin by providing a concise introduction to the changing dynamics in the field of intrusion detection.
- Then identify the four vital characteristics to be considered appropriately while devising network anomaly detection system.
- Propose anomaly detection system based on the suggested principles.

- Perform feature ranking and selection using information gain (IG) and automated branch-and-bound (ABB) algorithms, respectively.
- Implement logistic regression (LR) and eXtreme gradient boosting (XGBoost) techniques for classifying network traffic.
- Employ an emerging and powerful big data computing framework based on bulk synchronous parallel (BSP) processing.
- Evaluate the proposed system to verify its efficacy using ISCX-UNB dataset, which adequately represents network traffic patterns, and also highlight the significance of using appropriate datasets in anomaly detection domain.

The rest of the paper is organized as follows. The following section presents the background of network intrusion detection and related work. Section 3 introduces the BSP model, the big data framework Apache Hama, and also explains the motivation behind utilizing BSP based machine learning techniques for anomaly detection. Section 4 provides the architectural details of the proposed system. Section 5 presents the implementation and evaluation details followed by conclusion in Section 6.

2. Background and Related Work

Intrusion detection systems have been developed for over three decades, whereas the notion of big data trend in network security is relatively new. In order to establish a better understanding, in this section, we review some important concepts of network intrusion detection, related work and associated challenges that form the basis for motivation behind utilizing BSP-based machine learning computing in this area.

The possibility of automatic intrusion detection was first put forward by James Anderson in 1980 [17] in his classic paper, which states that a certain class of intruders or masqueraders who usually operate with stolen identities could probably be detected by their departures from the set norm for the original user. The proposed idea was basically to monitor security threats to audit trails. Afterward, the notion of checking all activities against a set security policy was introduced. Since Anderson's paper, numerous theories, methods, and hardware and software frameworks have been presented in the literature as well as in the form of commercial products. Consequently, security mechanisms such as firewalls, access control, and cryptography are now available and function as the first line of security defense with challenges involved from ever-evolving intrusion skills and techniques [18]. A firewall mainly protects the network resources by allowing and disallowing certain types of access on the basis of a configured security policy. An access control is usually deployed for authentication purposes, whereas cryptography is used to achieve secure communication. These traditional defensive techniques have several limitations in fully protecting networks and systems from increasingly sophisticated attacks and malware. Moreover, most systems built on such techniques suffer from high false positive and false negative detection rates and also lack the ability to continuously adapt with the changing malicious behaviors [18]. IDSs overcome certain limitations and provide a better security solution by protecting the network from both internal and external attacks. Table 2 summarizes the prominent network security solutions based on their significant characteristics.

Although both firewalls and IDSs are common network security solutions, they have significant differences in operation and functions. Traditional firewalls sniff out network packets at the network boundaries and are unable to detect complex attacks such as denial of service (DoS), distributed denial of service (DDoS), flooding attacks, user-to-root attacks, and port scanning. Moreover, firewalls have no intelligent way of identifying whether the network traffic is legit and normal [19]. An IDS diminishes threat impact and addresses such problems by performing an in-depth analysis of the network streams. It provides a more comprehensive defense against those threats and enhances network security. Access control and cryptography, on the other hand are more focused to ensure both confidentiality and integrity. We refer the reader to Axelsson [20] for a comprehensive detail on the taxonomy of IDSs.

Table 2. Common solutions to network intrusions.

Solution	Description	Intrusion Scope	Attack Types
Firewall	A system designed to stop unauthorized access.	External	IP spoofing, eavesdropping, DoS, port scan, and fragmentation attacks
Access control	A system that controls or limit illegal access.	External	Unauthorized access, password attacks, dictionary attacks, rainbow table attacks, and sniffer attacks
Cryptography	To stop the coding or decoding of secret messages.	External	Man-in-the-middle attacks, brute force attacks, and birthday attacks
IDS	A system that controls and monitors a network or a system.	Internal & External	DoS, DDoS, U2R, port scanning, and flooding

Notes: DoS = Denial of service; DDoS = distributed denial of service; U2R = user to root; IP = internet protocol; IDS = Intrusion detection system.

The issues discussed earlier have jolted the anomaly detection domain acutely, specifically the ones highlighted in Table 1. In order to cope with them, there is a great need to employ efficient big data technologies having real-time processing capabilities. Research efforts are afoot to address these problems by devising techniques using frameworks like Hadoop [21], Spark [22], and Storm [23] ecosystems. However, they are still in their infancy and, unfortunately most existing works are based on outdated datasets namely DARPA [24], University of California, Irvine, Knowledge Discovery and Data Mining Archive, 1999 (KDD Cup 99) [25], and its variations [26] that significantly reduce the value of such contributions. We address the issues related with evaluations based on these inadequate datasets later in this section.

Recently, Manzoor et al. [13] proposed network intrusion detection system using support vector machines (SVM) to classify incoming network traffic into benign or malicious. The authors utilized Apache Storm to handle computational requirements for big data networks. It is a development platform generally used to develop real-time big data stream processing applications. In this work, the proposed storm topology consist of one spout and three bolts: (i) input reader, the only spout which reads a network trace and forwards it to the next bolt; (ii) data preprocessor, which mainly performs data conversion and normalization functions; (iii) an SVM algorithm, which performs the classification task; and (iv) result aggregator is the last bolt which aggregates the classification results and store them in a file. Evaluations have been performed using KDD Cup 99 datasets; however a detailed experimental analysis and some important performance metrics are missing in this work.

Rathore et al. [14] proposed a real-time intrusion detection system in a ultra high-speed big data environment using the Hadoop framework. The architecture of the proposed system consists of four layers: (i) traffic capturing, which reads the network traffic; (ii) filtration and load balancing server, which performs filtration of the network traffic to achieve preliminary flow identification using in-memory database and also achieve load balancing of network packets among master and slave nodes using IP addresses; (iii) the processing layer, which is an important component of the system composed of various master and data nodes of Hadoop. Moreover, the authors also briefly claimed the use of Apache Spark to achieve real-time processing capabilities, however its use has not been justified well and (iv) the decision server layer, which performs classification operations using machine learning algorithms such as J48, Reduced Error Pruning Tree (REPTree), and Support Vector Machine (SVM). The evaluations are performed using DARPA, KDD Cup 99, and NSL_KDD datasets.

Janeja et al. [15] presented the architecture of a Big-distributed intrusion detection system (B-dIDS) to detect multipronged attacks existing across multiple subnets in a distributed network. The proposed solution is composed of two main components: a big data processing framework named HAMR, developed by HAMR Analytic Technologies (HAMRTECH), and an analytics engine consisting of a novel ensemble algorithm. Basically, the authors used an ensemble classification technique to automatically classify large amount of network data and to provide summarized alerts to system

administrator for possible malicious activities. This study motivates the need for employing specialized tools to implement intrusion detection systems, however lacks in performing benchmark evaluations.

Similarly, the studies in [27–31] also addressed the notion of intrusion detection in large-scale networks and paved the path forward by devising various strategies and architectures. In [27], the authors proposed centralized parallel Snort-based NIDSs to deal with the issues of high speed networks using various multicore processors and operating systems. The system attempts to enhance the performance in terms of reducing packet drop ratio, and it also helps the network security management to keep track of all attack behaviors to develop and enhance security policies. In [28], the authors focused on NIDSs' weaknesses in terms of their inability to perform efficient packet processing in high speed networks. The proposed approach adopted quality of service configuration and parallel technologies in Cisco Catalyst Switches to improve the performance of a NIDS and to reduce the packet drop ratio that may be caused by several types of attacks. Similarly, Vasiliadis et al. [29] addressed the challenges of intrusion detection in large-scale networks and presented a multiparallel intrusion detection architecture to cope with the increased processing throughput requirements.

Despite significant contributions, there is still need to devise more efficient solutions by employing appropriate computing technologies, robust machine learning techniques, and adequate datasets as briefly indicated earlier. Therefore, the principle novelty of this paper lies with the use of BSP-based machine learning techniques, specifically XGBoost, with cutting edge real-time big database (i.e., ISCX-UNB). Regarding the role of datasets for NIDS evaluation, the extensive use of nearly two decades of the old KDD dataset family in this modern era is disappointing, specifically when some better alternatives are available [16,32–34]. These flawed datasets lack veracity from the perspective of big data as indicated in Table 1, and so they would not be relevant as a consequence of having poor quality. Due to this low veracity, these datasets would also lack value as well, further reducing their relevancy. This deduces that the performance evaluation of a system is always considered of fundamental importance and must be based on standardized approaches; especially it is much vital to evaluate IDS with contemporary datasets. Failing to do so does not contribute to IDS research even if the system reliably detects all attacks showing the best possible capabilities. It has also been reported that the results of such contributions become irrelevant mostly when performed cross-validation with contemporary workloads [35]. Therefore, the models and evaluations based exclusively on outdated datasets place a shadow of doubt on their usefulness and also impair the ability to explore further knowledge horizons.

3. Utilizing Machine Learning and Bulk Synchronous Parallel Computing Techniques

Intrusion detection has led to a big data problem mainly due to the nature of its requirement to perform massive data analysis. The integration of machine learning techniques and big data processing technologies have helped in correctly identifying malicious activities in IDSs, which in turn help to improve network security. The tasks to generate efficient algorithmic techniques and selecting an appropriate computing framework are equally important in order to build an efficient IDS. Despite great efforts, some issues regarding their selection need further attention that could be comprehended with the following scenarios.

The Cloud Security Alliance reported in 2013 that an enterprise such as Hewlett-Packard (HP) can generate one trillion events per day or about 12 million events per second [36]. It was also reported that such large volumes of data are overwhelming and even data storage is becoming the primary concern rather than processing. It became very difficult for enterprises to rely on traditional software applications to deal with such big data issues. Moreover, traditional platforms like relational databases do not scale effectively against the onslaught of big data challenges posed by intrusion detection. Therefore, enterprises at large scale started to utilize big data solutions like Hadoop to better accommodate the storage requirements of massive volume with potentially very diverse heterogeneous data structures. Similarly, attempts have been made to make use of Hadoop in developing anomaly

detection systems which is not very suitable to employ for such applications. These systems heavily depend on iterative and real-time capabilities whereas a map-shuffle-reduce paradigm of Hadoop often becomes a bottleneck to address such challenges [37]. As such, the focus has been shifted to develop IDSs using frameworks like Apache Spark and Apache Storm etc. In this work, we use a BSP-based processing model named Apache Hama [38] to develop anomaly detection system using efficient machine learning techniques.

Hama, a top level project of Apache Software Foundation, is a distributed computing framework based purely on the BSP programming model, which acts as a bridge between software and hardware for parallel computing [39]. In the BSP model, the input data is divided between concurrently working tasks and the computation process consists of a sequence of iterative supersteps separated by barrier synchronization, as shown in Figure 1a. This methodology facilitates programmers by reducing the overhead of managing memory in performing local computations, manipulating global communications, and implementing efficient barrier synchronization. Figure 1b illustrates the process of computing the maximum vertex value using a BSP programming technique. The overall computation process in a BSP-based solution involves multiple supersteps and each superstep is divided into three smaller steps: local computation, global communication, and barrier synchronization. In the local computation step, a user-defined function is applied to every subtask concurrently, which effectively parallelizes the computation for 'read' and 'write' operations. In the global communication step, processes exchange their locally produced data according to the requests made during the local computation phase. Finally, the barrier synchronization phase ensures completion of all communication actions and makes previously exchanged data available to processes for use in the next superstep.

Among big data processing frameworks used in practice, Hama follows the original BSP model relatively closely [40,41]. It supports diverse massive computational tasks in several application domains such as graphs, matrices, machine learning, deep learning, and network algorithms. Like many other frameworks, it utilizes Hadoop Distributed File System (HDFS) as underlying environment as shown in Figure 1c. However, it is significantly different from other frameworks, incorporating versatile graph processing techniques and a wide range of application domains. It has the capability to significantly outperform MapReduce-based frameworks including the BSP-inspired framework, Giraph [41,42]. This work introduces another landmark of a BSP model to develop an anomaly detection system using parametric and nonparametric classification schemes, namely logistic regression [43] and XGBoost [44] respectively. The results obtained in Section 5 demonstrate the viability of BSP model also in the field of intrusion detection.

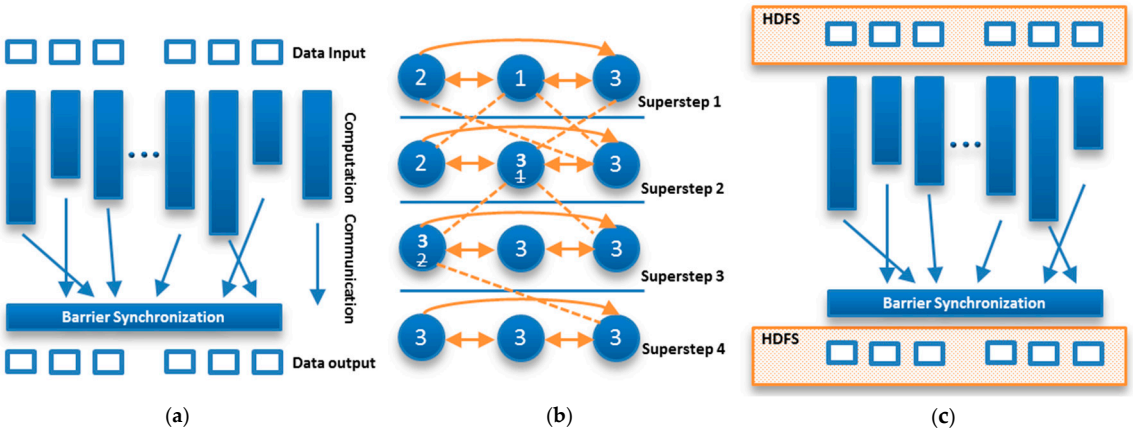

Figure 1. Work flow of the bulk synchronous parallel (BSP) model and Apache Hama: (**a**) BSP Model; (**b**) A BSP computing example; (**c**) Hama BSP framework.

4. Proposed Framework

The architecture of the proposed framework is depicted in Figure 2. Basically it involves three major phases: (i) input, which captures the network traffic and transmits it to the next phase; (ii) analysis, which is the most significant phase of the system where actual computation is performed and it consists of several components; and (iii) output, where alerts are being generated to identify malicious activities. The following subsections explain each phase in detail.

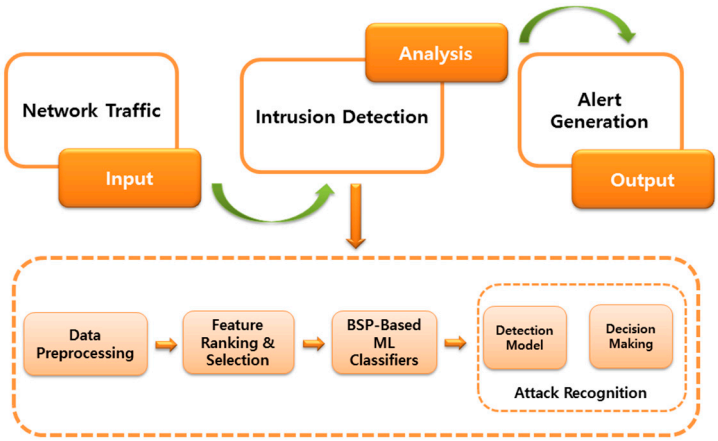

Figure 2. The architecture of the proposed intrusion detection system.

4.1. Input

It is the basic but most significant part of any system that greatly affects its performance and operations. In the case of IDSs, these are network flows in real-time environments or may be recorded network traces often called workloads or datasets. The type, quality and the location where data is collected from are the determinate factors in the design and effectiveness of an IDS. We believe that the productivity of NIDS research is largely dependent on the quality of datasets being used in addition to computational techniques involved. Based on these principles, we decided to use the ISCX-UNB dataset [16] as input to our proposed system for further experimental evaluations. The dataset details are given in Section 5.1.

4.2. Analysis

This is the core of the proposed system which performs an in-depth analysis of the network traces based on the following components.

4.2.1. Data Preprocessing

This component is responsible to preprocess the data involving conversion and normalization operations in order to make it ready for feature ranking and selection. The dataset consists of miscellaneous types of data including symbolic and numeric representation such as *protocolName* and *totalDestinationpackets*, respectively. The classification techniques require each record in the input data to be represented as a vector of real numbers; therefore, every symbolic feature in a dataset is first converted to a numerical value in this phase. The integers from 0 to $N - 1$, where N is the number of symbols, are assigned to each symbolic feature and then each value is linearly scaled to the range of [0–1]. For instance, the three symbols in *protocolName*—TCP, UDP, and ICMP—are assigned the values of 0, 1 and 2 respectively and are then scaled to 0, 1/3 and 2/3. Logarithmic scaling with base 10 has been applied to numeric features having a large integer value ranges like *totalDestinationPackets* and *totalSourcePackets* to reduce value ranges. This process brings dispersion in smaller values and compression in larger values, and hence interpretation becomes easier without losing information.

Once the conversion is done, an essential step of data normalization is performed to avoid the biasing factor in favor of features with greater values. It is a process of scaling the value of each attribute into a well-proportioned range. Every feature within each record is normalized by the respective maximum value and falls into the same range of [0–1].

4.2.2. Feature Ranking and Selection

One of the important factors in developing an intrusion detection system, which uses machine learning technique, is to devise an optimal feature selection scheme that reliably detects malicious activities. Although machine learning has been introduced in IDS to deal with finding patterns in big data, the presence of redundant features may degrade the classification performance significantly and pull the efficiency of learning algorithms down if they were not properly excluded. Moreover, it increases the computational resource needed during training and testing of IDS models. An optimal feature selection scheme helps to reduce computation time, improve prediction performance, and understand the data in in developing IDSs. To accomplish this task, we first applied information gain measure to rank the features based on their importance and then utilized automated branch-and-bound technique [45] to obtain optimal feature subset.

Information gain evaluates the worth of an attribute by measuring the IG with respect to the class. It has a tendency to choose features with more distinct values. It is based on the concept of entropy which is widely used in the information theory domain. If data D is split by feature X into p partitions, and d classes given by c_i, the information for D is defined as:

$$I(D) = -\sum_{i=1}^{d} P_D(c_i) log_2 P_D(c_i), \tag{1}$$

where the information for D_j due to partition D at X is defined as,

$$I({D_j}^X) = -\sum_{i=1}^{d} P_{D_jX}(c_i) log_2 P_{D_jX}(c_i), \tag{2}$$

And the information gain for the feature X is obtained as,

$$IG(X) = I(D) - \sum_{j=1}^{p} \frac{|D_j|}{|D|} I({D_j}^X), \tag{3}$$

where $|D|$ is the number of instances in D, and $P_D(c_i)$ represent the prior probabilities for data D estimated by $P(c_i) = |c_{i,D}|/|D|$. The time complexity to retrieve list of ranked features is $O(N^2)$, where N is the number of features.

Figure 3 shows the procedure of feature selection whereas the process of IG evaluation and feature selection using ABB technique is defined using Equations (1)–(3) and Algorithm 1 respectively [45]. Unlike a conventional algorithm of branch-and-bound in which the bound is predetermined, ABB determines it automatically. This algorithm starts with reading a complete set of features and attempts to remove one feature at a time using a breath-first search mechanism until base criteria is satisfied. In this process, a node acts as a subset of features and a legitimacy test for each node is performed to ensure valid execution. A node is said to be legitimate if the Hamming distance between the node being visited and a pruned node is not 1. Since one feature is removed in each iteration whose time complexity is $O(N)$, where N is the number of features. As a matter of fact, m iterates are required to complete the procedure. Therefore, the overall time complexity of ABB is $O(mN)$, where m is the maximum number of children a node can have and it is always smaller than N. The final integrated results of both IG and Algorithm 1 provided us with the best feature subset to be used for classification techniques. The details of selected features to be used for the proposed system are presented in Table 3.

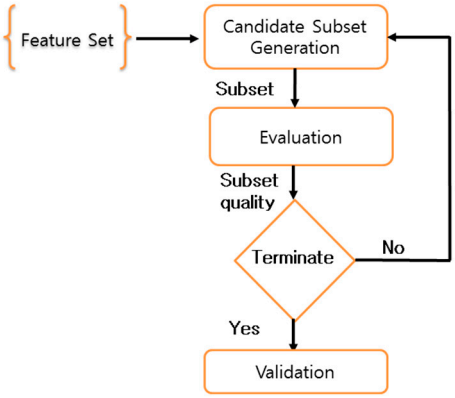

Figure 3. Feature selection process.

Algorithm 1. Automated branch-and-bound feature selection

Input: S—Training dataset D with features Xi
where i = 1, 2, 3 . . . N
Q—An empty queue, S1, S2—temporary subsets
U—Evaluation measure (inconsistency)
Output: A selected feature subset S
1: initialize L = {S}
2: α = U (S2, D)
2: ABB(S, D)/*Main function reading all features in data D */
3: do
4: S1 = S − X;
5: add S1 to Q;
6: while Q is NOT empty
7: S2 = delete from Q;
8: if (S2 is legitimate and U (S2, D) $\leq \alpha$)
9 Append S2 in L;
10 ABB (S2, D)
11: while (i = 1 to N);
12: Return the minimum subset;

Table 3. Selected features for the proposed system.

Serial No.	Feature Rank	Feature Name	Type	Description
1	f5	totalSourcePackets	Numeric	Total number of packets transmitted from source to destination
2	f4	totalDestinationPackets	Numeric	Total number of packets transmitted from destination to source
3	f9	direction	Text	Direction of the flow e.g., L2L, L2R etc.
4	f13	protocolName	Text	Type of the protocol, e.g., tcp, udp, etc.
5	f12	source	Text	Source IP
6	f15	destination	Text	Destination IP
7	f17	startDateTime	Date	Start timestamp of the connection
8	f18	stopDateTime	Date	Stop timestamp of the connection

4.2.3. BSP-based Machine Learning Classifiers

Once the optimal subset of features is obtained, it is taken as input to the classifier training phase where we employ two efficient machine learning classifiers namely, logistic regression and XGBoost.

Logistic regression, also known as logit regression or logit model, is a function-based classifier that uses ridge estimator and considered suitable approach to binary classification problems [43].

The binary logistic model determines the relationship between binary outcomes and independent variables by using a probability as the predicted value of the dependent variable. The binary outcomes specify the presence (1) or absence (0) of a certain characteristic or outcome in general. Given a feature X, it attempts to find whether a particular activity Y exists or not. For example, if an activity exists, Y is assigned a value of 1 and 0 otherwise. In the case of anomaly detection, the value of Y represents whether the network traffic flow is malicious or normal. Each flow or record is assigned a probability which is then used to classify it as anomalous or benign. This model generates a logistic curve and the predicted probability must lie between 0 and 1. This is where simple linear regression techniques are inefficient to perform because they allow the dependent variable to pass these limits and to produce inconsistent results. Defining the probability of an object belonging to group 1 as P_1, and P_0 for group 0, the logistic regression model can be written as:

$$Z_i = \log(P_{i1}/P_{i0}) = b_0 + b_1 x_{i1} + b_2 x_{i2} + \ldots + b_k x_{ik}, \quad (4)$$

where P_{i1}/P_{i0} is called the odd ratio, k is the number of parameters, b_j represents the value of the j^{th} co-efficient, $j = 1, 2, \ldots, k$, and x_{ij} holds the value of the i^{th} case of the j^{th} predictor. The parameters (b_0 to b_k) of the logistic model are estimated via maximum likelihood method [46]. Finally, the probability of occurrence of an event as defined above is calculated as follows:

$$P(Y_i = 1 | X_i) = \frac{e^{b^T X_i}}{1 + (e^{b^T X_i})} = \frac{1}{1 + e^{-b^T X_i}}, \quad (5)$$

where Y_i is a dependent variable under study, and $e^{b^T X_i}$ is a linear predictor of the logistic regression function. The flows are classified as anomalous if $P_1 > 0.5$ and benign if $P_1 < 0.5$.

XGBoost is another method we employed to achieve classification tasks [44]. It is an efficient and scalable implementation of gradient boosting framework originally proposed by Friedman [47]. Among the library of machine learning algorithms, Gradient Tree Boosting (GTB) is a technique that dominates in several applications. Undoubtedly, it has provided state-of-the-art results on many standard classification benchmarks [48]. The motivation behind boosting was to build a procedure combining the outputs of many weak classifiers to produce a strong and robust classifier. In practice, a weak learner or classifier is a prediction model having relatively poor performance statistics such as detection rate or accuracy that often leads to unreliable conclusions, thereby making it impractical due to high rate of misclassification error. In order to turn weak classifier to a stronger one that could act as a powerful committee, the predictions of a number of independent weak learners need to be combined. This process is generally accomplished by taking into account the maximum frequency of each prediction of all weak learners as a final prediction. This may also be achieved by making use of the weighted average.

XGBoost is a new technique among the representatives of the boosting family such as Adaptive Boosting (AdaBoost) and GTB. Even though XGBoost is built on the idea of basic gradient boosting, it takes significantly less compute effort and produces much smaller models. In this work, we used *dmlc/xgboost* package, which provides interfaces for most popular data science languages like R, Java, Scala, C++ and Python [49]. The impact of this robust technique has been widely recognized in a number of machine learning and data mining challenges. Among the 29 challenge winning solutions published at Kaggle's blog during 2015, 17 solutions employed XGBoost. However, its application in the field of intrusion detection has not been seen before, and this study applies XGBoost to perform classification on a contemporary dataset.

4.2.4. Attack Recognition

Once the classifier is trained using the selected subset of features, the normal and intrusive data can be recognized by using the saved trained classifier. The test data is then transported to the saved

trained model to detect intrusions. The traces matching to the normal class are treated as normal data while others are reported as intrusive activities.

4.3. Output

The final phase of the proposed system acts like an output component of the traditional system which is responsible to present processed data in usable format, i.e., it mainly interacts with the user by generating alerts.

5. Implementation and Evaluation

5.1. Dataset and Experimental Setup

Considering the significance of using appropriate datasets for evaluating NIDSs as discussed earlier in this paper, we preferred to use ISCX-UNB datasets rather than following traditional approach to use legacy KDD dataset family. It was built on the concept of profiles to reflect network traffic and intrusions. The traces were obtained in seven days under practical and systematic conditions. It is a labeled dataset consisting of normal and malicious flows consisting of 2,381,532 and 68,792 records respectively. Furthermore, various multistage attack scenarios have been executed to generate malicious traces such as infiltrating the network from the inside: HTTP, DoS, DDoS using an Internet Relay Chat (IRC) botnet, and brute force Secure Shell (SSH).

The experiments were conducted on an Intel core i7-6500U CPU @2.5 GHz with 512 GB SSD, 8 GB RAM with Apache Hama version 0.7.1 installed on Ubuntu 12.04.

5.2. Performance Evaluation

To validate the proposed system performance, we used the following performance metrics.

- Accuracy has traditionally been considered the most important performance evaluation metric. In the proposed system, accuracy was expressed as the percentage of true IDS predictions,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{6}$$

  where TP (true positive) is the number intrusions/anomalies identified correctly, i.e., successfully identifying an attack; TN (true negative) is the number of normal flows identified correctly, i.e., successfully identifying acceptable behavior; FP (false positive) is the number of normal flows incorrectly classified as intrusions, i.e., false alarms; FN (false negative) is the number of intrusions/anomalies incorrectly classified as benign, the most serious and dangerous state: the IDS failed to detect an attack.
- Detection rate (DR) represents the percentage of correctly classified intrusions or attacks compared with the total number of intrusions,

$$DR = \frac{TP}{TP + FN}. \tag{7}$$

- False positive rate (FPR) represents the percentage of normal flows incorrectly classified as intrusions compared with the total number of normal flows,

$$FPR = \frac{FP}{FP + TN}. \tag{8}$$

5.3. Results and Discussion

Several experiments have been conducted to evaluate the performance and efficacy of the proposed system. For this purpose, accuracy, detection rate, and false positive rate metrics are

applied, as defined in Equations (6)–(8) respectively. The performance of the proposed system is shown in Table 4. Overall performance under both classification schemes is promising, obtaining at least 99.15% accuracy for the ISCX-UNB-Saturday dataset using logistic regression. Also, it is worth noticing that XGBoost performs comparatively better than logistic regression. Further, several observation can be extracted in the Table 4 such as the logistic regression classifier shows highest accuracy for the network flows named ISCX-UNB-Monday, which contains HTTP DoS attacks. Whereas it performs lower for ISCX-UNB-Tuesday, which contains DDoS using Internet Relay Chat (IRC) botnet attacks.

Table 4. Performance results of the proposed system.

Dataset/Network Flows	Logistic Regression			XGBoost		
	DR	FPR	Accuracy	DR	FPR	Accuracy
ISCX-UNB-Saturday	98.09	0.18	99.15	99.49	0.13	99.78
ISCX-UNB-Monday	99.39	0.58	99.53	99.37	0.35	99.34
ISCX-UNB-Tuesday	98.56	0.67	98.99	98.99	0.29	99.69
ISCX-UNB-Wednesday	99.11	0.45	99.26	99.48	0.58	99.68
ISCX-UNB-Thursday	99.23	0.39	99.44	99.69	0.16	99.79

Notes: Accuracy, DR, and FPR as defined in Equations (6)–(8), respectively.

The XGBoost scheme shows highest accuracy for the network flows named ISCX-UNB-Thursday, which contains brute force SSH attacks, whereas it performs slightly lower in detecting DOS attacks. The average accuracy rate for logistic regression and XGBoost is 99.27% and 99.65% respectively, with low false positive rates. In addition to the performance metrics given in Table 4, the time efficiency of the proposed system has also been measured. The metric we used is the training time, which is defined as the time taken to build the training model. Figure 4 illustrates the time taken by each classification schemes for about 120,000 network packets. It can be seen that the proposed system is also efficient in terms of low computational cost, specifically when utilizing XGBoost. In particular, the training time of XGBoost classifier is 1.01–1.18× better than the logistic regression scheme.

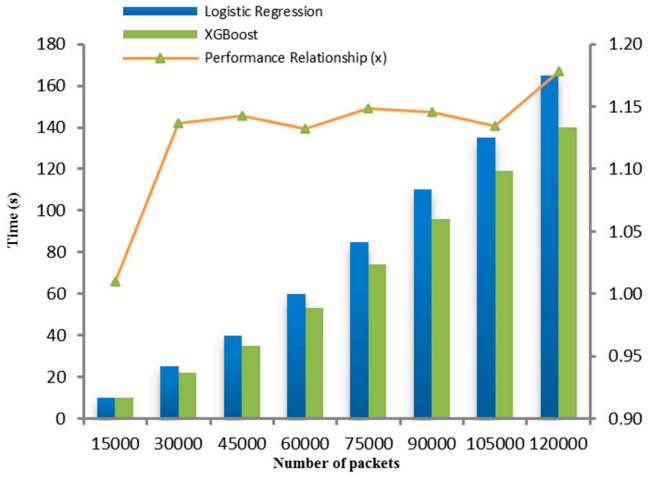

Figure 4. Training time for logistic regression and XGBoost schemes.

6. Conclusions

Anomaly detection is a significant issue in computer networks. The advances in high-speed big data networks and the concomitant rise in network attacks require fundamental methodological revisions to develop efficient NIDSs. Based on this need, we first identified four vital characteristics that form the basis to devise a comprehensive network anomaly detection system. The first two specifically deal with implementing machine learning concepts and greatly affect the performance

of the system, namely feature selection and the employment of classification schemes. The other two characteristics combat the challenges introduced by large-scale networks and sophisticated network attacks, namely utilizing specialized big data computing engines and obtaining contemporary workloads to conduct performance evaluations of the proposed systems. Building on the principles and issues analyzed in this paper, we proposed an efficient intrusion detection system that comprehensively follows the identified characteristics with an emphasis to handle big data problems in high speed networks. This is because anomaly detection in big data networks is particularly crucial due to the volume, velocity, variety, and veracity of the datasets. Thus, the proposed system incorporated a powerful BSP computing engine, which is capable of handling large volume of network traffic in real-time environments. Apart from using a real-time contemporary dataset for performing experimental evaluation, we also emphasized the role and importance of using appropriate datasets, which greatly lacks in existing studies. The experimental results on contemporary dataset verify the accuracy and efficiency of the proposed system. The future works include devising more novel NIDSs techniques using deep learning for more generalized (e.g., cross-scenarios) systems and higher security applications.

Acknowledgments: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program IITP-2017-2016-0-00465) supervised by the IITP (Institute for Information & communications Technology Promotion), and also supported by the "Building and Services of Information Security System Based on Advanced KREONET (Korea Research Environment Open NETwork)" program funded by the Ministry of Science, ICT & Future Planning (K-17-L01-C04-S03).

Author Contributions: Kamran Siddique conceived the research idea, built the prototype and performed the experiments. Zahid Akhtar and Haeng-gon Lee improved the working of the framework and also contributed in the background study. Kamran Siddique wrote the first draft of the paper. Yangwoo Kim, Woongsup Kim, and Zahid Akhtar assisted with revisions and improvements.

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

References

1. Heady, R.; Luger, G.F.; Maccabe, A.; Servilla, M. *The Architecture of a Network Level Intrusion Detection System*; Technical Report; Department of Computer Science, College of Engineering, University of New Mexico: Albuquerque, NM, USA, 15 August 1990.
2. Kim, D.S.; Park, J.S. Network-based intrusion detection with support vector machines. In *Information Networking*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 747–756.
3. Tsai, C.F.; Hsu, Y.F.; Lin, C.Y.; Lin, W.Y. Intrusion detection by machine learning: A review. *Expert Syst. Appl.* **2009**, *36*, 11994–12000. [CrossRef]
4. Kim, D.Y.; Jeong, Y.S.; Kim, S. Data-filtering system to avoid total data distortion in IoT networking. *Symmetry* **2017**, *9*, 16. [CrossRef]
5. 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.* **2013**, *5*, 75. [CrossRef]
6. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [CrossRef]
7. Suthaharan, S. Big data classification: Problems and challenges in network intrusion prediction with machine learning. *ACM SIGMETRICS Perform. Eval. Rev.* **2014**, *41*, 70–73. [CrossRef]
8. Whitworth, J.; Suthaharan, S. Security problems and challenges in a machine learning-based hybrid big data processing network systems. *ACM SIGMETRICS Perform. Eval. Rev.* **2014**, *41*, 82–85. [CrossRef]
9. Lee, Y.; Lee, Y. Toward scalable internet traffic measurement and analysis with hadoop. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 5–13. [CrossRef]
10. Grahn, K.; Westerlund, M.; Pulkkis, G. Analytics for Network Security: A Survey and Taxonomy. In *Information Fusion for Cyber-Security Analytics*; Springer: New York, NY, USA, 2017; pp. 175–193.
11. Wang, L.; Jones, R. Big data analytics for network intrusion detection: A survey. *Int. J. Netw. Commun.* **2017**, *7*, 24–31.
12. Zikopoulos, P.; Eaton, C. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*; McGraw-Hill Osborne Media: New York, NY, USA, 2012.

13. Manzoor, M.A.; Morgan, Y. Network intrusion detection system using apache storm. *Adv. Sci. Technol. Eng. Syst. J.* **2017**, *2*, 812–818. [CrossRef]
14. Rathore, M.M.; Ahmad, A.; Paul, A. Real time intrusion detection system for ultra-high-speed big data environments. *J. Supercomput.* **2016**, *72*, 3489–3510. [CrossRef]
15. Janeja, V.P.; Azari, A.; Namayanja, J.M.; Heilig, B. B-dids: Mining anomalies in a big-distributed intrusion detection system. In Proceedings of the 2014 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 27–30 October 2014; pp. 32–34.
16. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. [CrossRef]
17. Anderson, J.P. *Computer Security Threat Monitoring and Surveillance*; Technical Report; James P. Anderson Company: Fort Washington, PA, USA, 1980; Volume 17.
18. Pontarelli, S.; Bianchi, G.; Teofili, S. Traffic-aware design of a high-speed FPGA network intrusion detection system. *IEEE Trans. Comput.* **2013**, *62*, 2322–2334. [CrossRef]
19. Asosheh, A.; Ramezani, N. A comprehensive taxonomy of DDOS attacks and defense mechanism applying in a smart classification. *WSEAS Trans. Comput.* **2008**, *7*, 281–290.
20. Axelsson, S. *Intrusion Detection Systems: A Survey and Taxonomy*; Technical Report; Department of Computer Engineering, Chalmers University of Technology: Göteborg, Sweden, March 2000; Volume 99.
21. Apache Hadoop. Available online: https://hadoop.apache.org/ (accessed on 16 August 2017).
22. Apache Spark. Available online: https://spark.apache.org/ (accessed on 16 August 2017).
23. Apache Storm. Available online: https://storm.apache.org/ (accessed on 16 August 2017).
24. DARPA Intrusion Detection Datasets. Available online: https://www.ll.mit.edu/ideval/data/index.html (accessed on 16 August 2017).
25. KDD Cup 1999 Data. Available online: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 16 August 2017).
26. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the CISDA 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009.
27. Karim, I.; Vien, Q.T.; Le, T.A.; Mapp, G. A comparative experimental design and performance analysis of Snort-based Intrusion Detection System in practical computer networks. *Computers* **2017**, *6*, 6. [CrossRef]
28. Bul'ajoul, W.; James, A.; Pannu, M. Improving network intrusion detection system performance through quality of service configuration and parallel technology. *J. Comput. Syst. Sci.* **2015**, *81*, 981–999. [CrossRef]
29. Vasiliadis, G.; Polychronakis, M.; Ioannidis, S. MIDeA: A multi-parallel intrusion detection architecture. In Proceedings of the 18th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 17–21 October 2011; pp. 297–308.
30. Tan, Z.; Nagar, U.T.; He, X.; Nanda, P.; Liu, R.P.; Wang, S.; Hu, J. Enhancing big data security with collaborative intrusion detection. *IEEE Cloud Comput.* **2014**, *1*, 27–33. [CrossRef]
31. Marchal, S.; Jiang, X.; State, R.; Engel, T. A big data architecture for large scale security monitoring. In Proceedings of the 2014 IEEE International Congress on Big Data (BigData Congress), Anchorage, AK, USA, 27 June–2 July 2014; pp. 56–63.
32. MAWI Working Group Traffic Archive. Available online: http://mawi.wide.ad.jp/mawi/ (accessed on 16 August 2017).
33. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Towards generating real-life datasets for network intrusion detection. *IJ Netw. Secur.* **2015**, *17*, 683–701.
34. The UNSW-NB15 Dataset. Available online: https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/ (accessed on 16 August 2017).
35. Moustafa, N.; Slay, J. The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Inf. Secur. J.* **2016**, *25*, 18–31. [CrossRef]
36. Big Data Working Group. Big Data Analytics for Security Intelligence. September 2013. Available online: https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Analytics_for_Security_Intelligence.pdf (accessed on 16 August 2017).
37. Kalavri, V.; Vlassov, V. Mapreduce: Limitations, optimizations and open issues. In Proceedings of the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Melbourne, VIC, Australia, 16–18 July 2013; pp. 1031–1038.

38. Apache Hama. Available online: https://hama.apache.org/ (accessed on 16 August 2017).
39. Valiant, L.G. A bridging model for parallel computation. *Commun. ACM* **1990**, *33*, 103–111. [CrossRef]
40. Siddique, K.; Akhtar, Z.; Kim, Y.; Jeong, Y.S.; Yoon, E.J. Investigating Apache Hama: A bulk synchronous parallel computing framework. *J. Supercomput.* **2017**, *73*, 1–16. [CrossRef]
41. Siddique, K.; Akhtar, Z.; Yoon, E.J.; Jeong, Y.S.; Dasgupta, D.; Kim, Y. Apache Hama: An emerging bulk synchronous parallel computing framework for big data applications. *IEEE Access* **2016**, *4*, 8879–8887. [CrossRef]
42. Jakovits, P.; Srirama, S.N.; Kromonov, I. Viability of the bulk synchronous parallel model for science on cloud. In Proceedings of the 2013 International Conference on High Performance Computing and Simulation (HPCS), Helsinki, Finland, 1–5 July 2013; pp. 41–48.
43. Hosmer, D.W., Jr.; Lemeshow, S.; Sturdivant, R.X. *Applied Logistic Regression*; John Wiley & Sons: Hoboken, NJ, USA, 2013; Volume 398.
44. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
45. Liu, H.; Motoda, H. *Instance Selection and Construction for Data Mining*; Kluwer Academic Publishers: Norwell, MA, USA, 2001.
46. James, G.; Witten, D.; Hastie, T. *An Introduction to Statistical Learning: With Applications in R*; Springer: New York, NY, USA, 2013; Volume 103.
47. Friedman, J. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
48. Li, P. Robust Logitboost and Adaptive Base Class (ABC) Logitboost. *arXiv*, **2012**.
49. XGBoost. Available online: https://github.com/dmlc/xgboost (accessed on 16 August 2017).