Intrusion Detection based on Graph Oriented Big Data Analytics

By Ahlem Abid & Farah Jemili

Abstract- Intrusion detection has been the subject of numerous studies in industry and academia. Still, Cybersecurity analysts always want greater precision and global threat analysis to secure their systems in cyberspace.

To improve the intrusion detection system, the visualization of the security events in the form of graphs and diagrams is significant to improve the accuracy of alerts.

In this paper, we propose an approach of an IDS based on cloud computing, big data technique and, using a machine learning graph algorithm which can detect in realtime different attacks as early as possible. We use the MAWI-Lab intrusion detection dataset. We choose Microsoft Azure as a unified cloud environment to load our dataset on Azure blob storage. We implement the k2 algorithm, which is a graphical machine learning algorithm to classify attacks.

Our system showed a great performance due to the graphical machine learning algorithm and Apache Spark structured streaming engine.

Keywords: intrusion detection; MAWI-Lab; apache spark streaming; microsoft azure cloud; graph; machine learning; k2 algorithm.

GJCST-D Classification: I.2.8
Intrusion Detection based on Graph Oriented Big Data Analytics

Ahlem Abid* & Farah Jemili*

Abstract: Intrusion detection has been the subject of numerous studies in industry and academia. Still, Cybersecurity analysts always want greater precision and global threat analysis to secure their systems in cyberspace.

To improve the intrusion detection system, the visualization of the security events in the form of graphs and diagrams is significant to improve the accuracy of alerts.

In this paper, we propose an approach of an IDS based on cloud computing, big data technique and, using a machine learning graph algorithm which can detect in real-time different attacks as early as possible. We use the MAWILab intrusion detection dataset. We choose Microsoft Azure as a unified cloud environment to load our dataset on Azure blob storage. We implement the k2 algorithm, which is a graphical machine learning algorithm to classify attacks.

Our system showed a great performance due to the graphical machine learning algorithm and Apache Spark structured streaming engine.

Keywords: intrusion detection; MAWILab; apache spark streaming; microsoft azure cloud; graph; machine learning; k2 algorithm.

I. Introduction

An IDS [1] is a security software or hardware system that monitors the network environment and identifies malicious or unusual actions on computer systems to allow for system security to be maintained. Given the importance of an intrusion detection system, Information Technology (IT) departments in organizations deploy the IDS to obtain information on potentially malicious activities occurring in their technological environment.

It also allows information to be transferred between departments and organizations in an increasingly secure and reliable manner. An IDS is an upgrade to other cyber security technologies such as firewalls, antivirus, etc. [2] According to University of Maryland [3], M Cukier says that the computers were attacked by hackers every 39 seconds on average 2.244 times a day and data breaches exposed 4.1 billion records in the first half of 2019. Cyber attacks come from multiple sources, 48% of malicious email attachments are office files [4], 34% of data breaches involved internal actors [5] and 92% of malware is delivered by email [6].

In 2019, 43% of cyber attacks involved small businesses [5] because 53% of companies had over 1000 sensitive files open to every employee, but only 5% of folders were properly protected [7]. According to IBM[8], the average cost of a data breach in 2019 was $3.9 million. This is a loss that no small or even medium-sized business would be able to sustain. That’s why, as reported by Forbes [9], 83% of enterprise workloads will move to the cloud by the year 2020, so there is a need to develop a new and efficient IDS. To improve the intrusion detection systems, the visualization of the security events in the form of graphs and diagrams is important to improve the accuracy of alerts.

For this purpose, we propose an approach of an IDS based on cloud computing, Big data technique, and using a graph machine learning algorithm that can identify and detect in real-time different attacks as early as possible. We use the MAWILab intrusion detection dataset. We choose Microsoft Azure as a unified cloud environment to load our dataset on Azure blob storage. We will implement the k2 algorithm, which is a graphical machine learning algorithm to classify attacks.

The rest of this paper is organized as follows. Section 2 discusses a review of the related works. Section 3 describes the cloud computing platform Microsoft Azure and its functionalities. Section 4 describes the big data framework Spark and its components. While section 5 describes the intrusion detection dataset used in our research. Moreover, Section 6 presents the proposed approach. The experimental results are presented in Section 7. Finally, Section 8 provides the conclusion of the paper and offers perspectives for future research.

II. Related Work and Background

Intrusion detection has been the subject of numerous studies. The purpose of this section is to present related works and present the methods used currently to detect intrusions. These works used Big Data frameworks with machine learning to detect intrusions. Some used stream processing engines and graph processing.

Mehdi Ezzarii et al. [10] described the intrusion detection system, which maximizes detection accuracy and minimizes false alarms. The authors discussed the genetic algorithm and how the system detects intrusions to ensure effective security. Sheeraz Niaz Lighari and Dil Muhammad Akbar Hussain [11] analyzed the kddcup’99 database with supervised and unsupervised machine learning algorithms using apache spark as a big data
analysis tool. They compared these algorithms based on three characteristics, such as training time, prediction time, and accuracy. The results showed that the naïve Bayes algorithm takes less time in training and prediction, while the k-means algorithm achieved a high accuracy.

Mondher Essid and Farah Jemili [12] presented a new approach based on the big data map reduce technique that combined the two KDD'99 and DARPA databases. Then they implemented the Bayesian networks, and the K2 algorithm using the WEKA tool to analyze the base. Thanks to the use of Bayesian networks these works showed nice performance in the detection phase and low false alarm.

Yao, H., Y. Liu and C. Fang [13] have proposed a new model based on big data analysis. The simulation results reveal that, compared to the k-means, decision tree and random forest algorithms, has a better performance, which can achieve a detection rate of 95.4% on normal data, 98.6% on Dos attacks, 93.9% on Probe attacks, 56.1% on U2R attacks and 77.2% on R2L attacks.

M. Elayni and F. Jemili [14] presented a method of forming and merging three unstructured and heterogeneous KDD99, DARPA 1998, and DARPA 1999 alert bases in a local environment with Mapreduce under Mongoddb and to analyze the dataset; they used a Bayesian network using the K2 algorithm implemented in WEKA. They were intended to increase the intrusion detection rate and minimize the false positive rate. Mylavarapu et al. [15] proposed a new hybrid intrusion detection system consisting of two neural networks. The CC4 Instant Neural Network acts as an anomaly-based detection for unknown attacks, and the Multi-Layer Perceptron neuronal network acts as an abuse-based detection for known attacks. According to the results of these two neural networks, the incoming data will be classified as "attack" or "normal" they found that the average accuracy of the hybrid detection system is 89%, with a false positive rate of 4.32%. The entire simulation was performed using an Apache Storm cluster with the ISCX 2012 dataset.

Gupta et al. [16] have implemented an intrusion detection framework based on Spark. To assess their performance, they used five ML algorithms on the NSL-KDD and DARPA KDD’99 datasets. Despite the use of Spark’s batch mode for their work, their results show that the random forest classifier gives the best accuracy while the naïve Bayes hold a faster training/prediction time.

Manzoor et al. [17] introduced a real-time intrusion detection system using the Apache Storm Framework. This system involves applying the SVM algorithm and evaluated using the KDD’99 dataset. The system can process up to 13600 packets in one second on a single machine with 92.60% accuracy on test data and has a normal and malicious data packet detection rate of 97.51%. The proposed system has a false positive rate of 0.35% and a false negative rate of 2.14%. Terzi et al. [18] created a new unsupervised intrusion detection approach using Apache Spark on the Microsoft Azure HD Insight cluster to gain scalable processing power. The new approach was tested on a CTU-13 dataset and reached a 96% accuracy rate. The results were visualized after the dimension reduction using Principal Component Analysis (PCA). Pallaprolu et al. [19] applied Apache Spark streaming to detect zero-day attacks. The proposed system is tested with the KNN algorithm, which showed an accuracy of 99.57% with a true positive rate (TPR) of 94% and a false positive rate (FPR) of 3%.

M. Hafsa et F. Jemili [20] proposed a real-time Intrusion Detection System. Their proposed system uses Apache Spark Structured Streaming to process and detect anomalies in real-time. They used the MAWILab dataset to evaluate the proposed system against cyber-threats. Based on experimentation, Their Spark-based IDS yields a 99.95% accuracy using a Decision Tree classifier and can also process more than 55,175 records in one second using only a two worker-nodes cluster.

Hao Zhang et al. [21] proposed a real-time intrusion detection system for high-speed network environments, which is implemented by a distributed Spark-based Random Forest detection model. The system simulation process is implemented using the CICIDS2017 intrusion dataset in a framework consisting of logstash, Kafka, and the distributed Spark cluster.

Experimental results show that the proposed detection model has a shorter detection time, achieves greater accuracy, and can perform real-time intrusion detection in a high-speed network environment.

Mustapha Belouch et al. [22] tested intrusion detection performance using four classification ML algorithms using Apache Spark and its Mlib library. They used UNSW-NB15 as the experimental dataset for intrusion detection, the results show that the Random Forest classification algorithm gave the best performance in terms of accuracy (97.49%), sensitivity (93.53%) and prediction time (0.08 seconds).

Aulia Essra et al. [23] used the Hierarchical Graph Neuron (HGN) algorithm as an extension of the Graph Neuron (GN) algorithm to classify intrusions in computer networks using the KDD Cup 99 dataset. The results showed that the HGN algorithm is promising and stable in the classification of intrusion attack models with an accuracy rate of 96.27%, a detection rate of 99.20%, a true negative rate of less than 15.73%, and a low false-positive rate of less than 0.80%.

Mingqiang Z et al. [24] presented a graph-based intrusion detection algorithm using a detection method based on the local deviation coefficient LDCGB. Compared to traditional clustering intrusion detection
algorithms, this algorithm does not require the initialization of the number of clusters.

LDCGB uses a graphical cluster algorithm to obtain an initial partition of the dataset that depends on the cluster precision parameter rather than the initial cluster number. The algorithm test is performed by the Kddcup’99 dataset. The results showed that the proposed algorithm could achieve satisfactory performance with a good detection rate (more than 92%) and a low false-positive rate (2%). Liu, R., & Zhu, Q. [25] proposed a method for detecting network anomalies (NAD-NNG) by combining with the idea of Natural Neighborhood Graph. The algorithm uses the natural neighbourhood graph to aggregate the normal data set. Also, the algorithm could obtain an adaptive percentage value $\beta$ to define the anomaly threshold. They used the KDDCUP’99 experimental dataset; the results showed that the proposed method can achieve a higher detection rate (91.98%) based on a tolerable false alarm rate (4.64%) compared to the other two NSM-KNN [26] and NADCP [27] algorithms.

Thang, V. V et al. have [28] introduced a new algorithm called Incrementalssgc, which is a semi-supervised clustering algorithm that was based on the graph of k-neighbor nearest using seeds. Experiments carried out on certain UCI data sets such as Iris, Wine, Ecoli, etc. and the AWID data set show the effectiveness of this new Incrementalssgc algorithm compared to other Incremental dbscan and SSGC algorithms.

Based on the methods of existing works, we propose a new system which uses an evolutive and updated evaluation dataset the MAWILab dataset and Also uses apache spark structured streaming hosted in a cloud environment to simulate real-world scenarios. A graphical machine learning algorithm k2 algorithm is used within a distributed environment implemented in Microsoft Azure Cloud. More details about Microsoft Azure are given in section 3.

III. MICROSOFT AZURE

Microsoft Windows Azure [29] is a public cloud computing platform that was available in 2010. Azure offers hybrid cloud solutions as well full support and integration with Microsoft products, which makes it a market leader and first direct competitor to Amazon Web Services.

Azure offers three main cloud computing platform services SaaS (Software as a Service), IaaS (Infrastructure as a Service) and PaaS (Platform as a Service). It lists over 600 Azure services. Ordering from storage, analytics, and virtual computing to big data enterprise management solutions, machine learning tools, and much more.

Microsoft's Azure Machine Learning [30] allows developers to write, test, and deploy algorithms, as well as access a marketplace for off-the-shelf APIs. Microsoft has designed Azure HDInsight [31] [32] based on the Hortonworks Data Platform (HDP) which contributes enormously to making HDInsight a robust data processing and analysis service. It was designed by Microsoft as a cloud-based service for processing and analysis of large volumes of and historical data. It enables developers to use and build big data applications using open source frameworks like Apache Hadoop, Apache Spark, Apache Hive, Apache Kafka,… HD Insight was designed to make Apache Hadoop and Apache Spark simple to use, with lower manageability costs and higher developer productivity. Customers have seen a 63% lower total cost of ownership (TCO) and 66% higher IT staff efficiencies by deploying HD Insight over on- premises Hadoop deployments[33].

IV. APACHE SPARK

a) Apache Spark

Apache Spark [34] is a powerful hybrid, fast, flexible, and fault-tolerant distributed data processing framework. It's the most active open source project in big data. Spark was originated at the University of California Berkeley in the AMP Lab in 2009.

Apache Spark in Azure HDInsight [35] is the Microsoft implementation of Apache Spark in the cloud. HDInsight makes it easier to create and configure a Spark cluster in Azure. Spark clusters in HDInsight are compatible with Azure Storage and Azure Data Lake Storage. So you can use HDInsight Spark clusters to process your data stored in Azure. It provides high-level APIs in Scala, Java, Python, and R languages.

Spark powers a stack of libraries including SQL and Data Frames, MLlib for machine learning, GraphX, and Spark Streaming (Figure 1).
All the functionalities being provided by Apache Spark are built on the top of Spark Core which is the foundation of parallel and distributed processing of huge datasets.

b) Spark Structured Streaming

Structured Streaming [36] [37] is another way to handle streaming with Spark. It was introduced in Apache Spark version 2.0. Scalable, fault-tolerant, like Spark streaming. Spark structured streaming allows the user to treat the data as if it were a table that would fill perpetually: each new occurrence in the stream could result in the addition of a new row in the table. Spark can process data flows using the same API as for batch processing.

This input table can then be transformed by a query that is run incrementally to calculate a result table:

- The user sets a trigger that will determine the time interval for updating the result table.
- When the results table is updated, these are persisted on external storage.

The way to persist the data is defined for an "output mode".

c) Machine Learning Pipeline

MLlib standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline.

The key concepts introduced by the Pipelines API are [39]:

Data Frame: This ML API uses Data Frame from Spark SQL as an ML dataset, which can hold a variety of data types.
Transformer: A Transformer is an algorithm that can transform one Data Frame into another Data Frame.

Estimator: An Estimator is an algorithm which can be fit on a Data Frame to produce a Transformer. A Pipeline is specified as a sequence of stages, and each stage is either a Transformer or an Estimator. These stages are run in order, and the input Data Frame is transformed as it passes through each stage.

If we want to experiment with a new idea or to use an existing technique that isn’t implemented in MLlib, we can extend Spark’s MLlib or Spark’s ML pipelines with our custom algorithms [40][41][42], but for Spark ML pipelines, we can lose some of the integrated properties of the pipeline, including the ability to automatically run meta-algorithms [43].

In our work, we will implement a graphical machine learning algorithm k2 in spark MLlib as a classification algorithm.

V. INTRUSION DETECTION DATASET

The evaluation datasets [44] play a vital role in the validation of any IDS approach by allowing us to assess the proposed method’s capability in detecting intrusive behavior. In the literature [45] [46], we find hundreds of publicly available datasets for intrusion detection, and they are widely used. We quote some of them DARPA, KDD’99, NSL KDD, MAWILAB, CTU-13, CICIDS 2017, UNSW-NB15 …

To evaluate our system, we will use the MAWILab dataset.

a) MAWILab Dataset

MAWILAB dataset [47] used to evaluate intrusion detection methods consists of packet traces from the MAWI archive. The dataset is updated daily to include new traffic from applications and anomalies: Since 2001 [48], Each trace in this database collects the traffic captured for 15 min in a specific day on a backbone link between Japan, and the USA. These data are collected by the Fukuda laboratory, and are published in two versions.

MAWILab contains ten fields: anomalyID, srcIP, srcPort, dstIP, dstPort, taxonomy, heuristic, distance, nbDetectors and label. Table 1 below shows these fields and their meanings.

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnomalyID</td>
<td>A unique anomaly identifier. This field permits to identify lines that refer to the same anomaly</td>
</tr>
<tr>
<td>SrcIP</td>
<td>The source IP address of the identified anomalous traffic</td>
</tr>
<tr>
<td>SrcPort</td>
<td>the source port of the identified anomalous traffic</td>
</tr>
<tr>
<td>DstIP</td>
<td>the destination IP address of the identified anomalous traffic</td>
</tr>
<tr>
<td>DstPort</td>
<td>the destination port of the identified anomalous traffic</td>
</tr>
<tr>
<td>Taxonomy</td>
<td>the category assigned to the anomaly using the taxonomy for back-bone traffic anomalies</td>
</tr>
<tr>
<td>Heuristic</td>
<td>the code assigned to the anomaly using simple heuristic based on port number, TCP flags and ICMP code</td>
</tr>
<tr>
<td>Distance</td>
<td>the difference between distance to normal traffic and distance to anomalous traffic</td>
</tr>
<tr>
<td>NbDetectors</td>
<td>The number of configurations that reported the anomaly.</td>
</tr>
<tr>
<td>Label</td>
<td>Assigned to the anomaly, it can be either: anomalous, suspicious, or notice.</td>
</tr>
</tbody>
</table>

The field label has four classes:
- Anomalous is assigned to all abnormal traffic and should be identified by any efficient anomaly detector.
- Suspicious is assigned to all traffic that is probably anomalous but not identified by our method.
- Notice is assigned to all traffic that is not identified anomalous by our method, but that has been reported by at least one anomaly detector.
- Benign all the other traffic is labeled because none of the anomaly detectors identified them.
Also, MAWILab employs two distinct anomaly classification techniques, a simple heuristic and a taxonomy [48] of backbone traffic anomalies based on protocol headers and connection patterns.

VI. Proposed Approach

In this work, we propose an approach based on graph model-oriented Big Data Analytics to classify intrusions in real-time (Figure 3). The proposed system has the following steps:

- Data Collection
- Data preprocessing
- Intrusion Detection based graph in real-time

a) Data Collection

This step consists on the collection of dataset files from the Fukuda laboratory website and uploads it into Azure Blob Storage. To obtain dataset files, first, we will be using the Microsoft Azure SDK for Python which provides a set of Python packages that make it easy to access the Microsoft Azure components such as storage, and Beautiful Soup[49] which is an incredible tool for parsing HTML and XML documents from a website.

The Fukuda website does not provide a single directory for published files, but each file is in a separate HTML page containing the CSV and XML files, Beautiful soup help us to extract the data easily directly from the Fukuda website. Since MAWILab dataset is updated daily, any new file added to their website will be immediately ingested by our web scraper[50].

b) Data Preprocessing

To obtain clean and usable data, we will transform and clean data by eliminating irrelevant features and redundant rows. Data preprocessing will be done using a Microsoft HDInsight cluster running Apache Spark Structured Streaming, which offers both distributed storage and processing also include a Jupyter notebook using the Python API of Spark (PySpark).

This step is divided into two parts:

i. Transforming Data

Because Apache Spark offers the capability to convert data from its original format to another format, in this step, we launch Jupyter Notebook and start by reading CSV files as a stream of data from the Azure Blob Storage[51] and convert them to Apache Parquet format [52].

ii. Cleaning Data

- Feature Selection

Feature selection [44] is helpful in decreasing the computational difficulty, eliminate data redundancy, enhance the detection rate of the machine learning techniques, simplify data, and reduce false alarms.

The alert database contains several redundant connections that have been captured multiple times, which poses a learning algorithm problem. Indeed, this problem causes an algorithm bias towards frequent registrations. This prevents the algorithm from learning rare recordings, which are usually more harmful...
to networks. This issue is mitigated by deleting all repeated records in the Mawilab Learning and Test Set by keeping only one copy of each record.

c) Intrusion Detection based graph in real-time

The Graphical model [53] is a branch of machine learning which uses a graph to represent a domain problem. Many machine learning algorithms belong to a graphical models, such as the Bayesian network algorithm, Naive Bayes algorithm, the Hidden Markov Model.

In our work, We implement Bayesian networks such as the K2 classification algorithm in Spark MLlib in a distributed manner to analyze our preprocessed data.

i. Bayesian Network

Several studies have demonstrated the performance of Bayesian networks in the field of intrusion detection.

A Bayesian network [54] is a probabilistic graphical model that uses Bayesian inference for probability calculations. This model aims to model conditional dependence and causality by representing conditional dependence by edges in a Directed acyclic graph (DAG) in which each edge corresponds to a conditional dependence, and each node corresponds to a single random variable.

Formally, if an edge (A, B) exists in the graph linking the random variables A and B, this means that $P(B / A)$ is a factor of the joint probability distribution, it is, therefore, necessary to know $P(B / A)$ for all values of B and A to make an inference. Bayesian Networks help us in decision making and simplify complex problems encoding various independencies.

ii. K2 Algorithm

The K2 learning algorithm has shown high performance in many research works. This algorithm was proposed by Cooper and Herskovits in 1992[55]. Its principle is as follows:

The K2 algorithm requires a well-defined order of variables. From this order, the K2 algorithm determines the dependency links between these attributes now called variables or nodes of the graph to be built. For each node, K2 tests its dependence against the preceding nodes in order and evaluated by the Bayesian measurement. If this dependence is verified, a new arc is added to the graph provided, it improves the Bayesian measurement and does not exceed the set limit number of emerging arcs to a node. The limit number of parents of a node ‘u’ required for algorithm input is set to $u = \text{number of variables} - 1 = 7$.

Note that this number takes a maximum value. This value is chosen in order to that the last node (Label), as the target variable, can have a maximum of 7 parents.

Below is the pseudo-code of the k2 algorithm:

```
1. procedure K2;
2. {Input: A set of $n$ nodes, an ordering on the nodes, an upper bound $u$ on the number of parents a node may have, and a database $D$ containing $m$ cases.}
3. {Output: For each node, a printout of the parents of the node.}
4. for $i=1$ to $n$ do
5. $\pi_i := \emptyset$;
6. $P_{\text{old}} := f(i, \pi_i)$; {This function is computed using Equation 1.}
7. OKToProceed := true;
8. While OKToProceed and $|\pi_i| < u$ do
9. let $z$ be the node in Pred($x_i$) - $\pi_i$ that maximizes $f(i, \pi_i \cup \{z\})$;
10. $P_{\text{new}} := f(i, \pi_i \cup \{z\})$;
11. if $P_{\text{new}} > P_{\text{old}}$ then
12. $P_{\text{old}} := P_{\text{new}}$;
13. $\pi_i := \pi_i \cup \{z\}$;
14. else OKToProceed := false;
15. end {while};
16. write(‘Node ‘, $x_i$ , ‘ Parent of ‘, $\pi_i$);
17. end {for};
18. end {K2};
```

**Figure 4:** K2 algorithm [56]

The equation is included below:

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_k - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}!$$
iii. **Real-time Detection based graph**

After cleaning data, we proceed by loading our machine learning pipeline model, which contains the K2 algorithm classifier, and we transform incoming data to obtain predictions for each record. We have created a machine learning model from the Fuduka Lab and in this task, we simply load the saved model, and test it against a live stream of files.

Spark Structured Streaming starts reading parquet files in a real-time manner, an acyclic-oriented graph resulting from the application of the K2 algorithm was displayed (figure 5).

![Figure 5: The Bayesian network resulting from the application of the K2 algorithm](image)

At each node of the graph, three measures are associated: a measure of probability, a measure of possibility, and a measure of necessity.

We transform the initial graph into a junction tree, then infer the observed evidences in the tree and deduce the evidences of the variable « Label ». Next step, show prediction results: for each value of the « Label » variable, three measurements are assigned: a probability measure, a possibility measure, and a necessity measure, display the outcome of the decision-making process to select the value of the « Label » variable with the highest informative probability and greater than one fixed threshold.

Figure 6 shows the result of the prediction and the outcome of the decision-making process:

![Figure 6: Prediction](image)
VII. EXPERIMENTATION AND RESULTS

a) IDS Evaluation

i. Performance Metrics

Intrusion detection systems are typically evaluated by the following standard performance measures [56] [57].

The metrics are listed in Table 2 given below. For each metric, a brief description and the calculating formula are given.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Rate or Accuracy (CR)</td>
<td>The ratio of correctly classified instances and the total number of instances.</td>
<td>CR = Correctly Classified instances / Total number of instances = TP+TN / TP+TN+FP+FN</td>
</tr>
<tr>
<td>Detection Rate (DR) or True Positive Rate (TPR)</td>
<td>The ratio between the number of correctly detected attacks and the total number of attacks.</td>
<td>DR = Correctly detected attacks / Total number of attacks = TP/TP+FN</td>
</tr>
<tr>
<td>False Positive Rate (FPR)</td>
<td>The ratio between the number of normal instances detected as attack and the total number of normal instances</td>
<td>FPR = Number of normal instances detected as attacks / Total number of normal instances = FP/FP+TN</td>
</tr>
<tr>
<td>Precision (PR)</td>
<td>It is the fraction of data instances predicted as positive that are actually positive.</td>
<td>PR = TP/TP+FP</td>
</tr>
<tr>
<td>Recall</td>
<td>Proportion of positive examples that were classified correctly</td>
<td>R = TP/TP+TN</td>
</tr>
<tr>
<td>F-Measure</td>
<td>The harmonic mean of the precision and recall</td>
<td>FM = 2 * PR * R / PR + R</td>
</tr>
</tbody>
</table>

ii. Results

Using the graphical algorithm k2 in this experiment showed great results as indicated in the following tables 3, 4, and 5.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Result (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>98.85</td>
</tr>
<tr>
<td>Precision</td>
<td>98.76</td>
</tr>
<tr>
<td>Recall</td>
<td>98.81</td>
</tr>
<tr>
<td>F-Measure</td>
<td>98.78</td>
</tr>
</tbody>
</table>

Table 3: Proposed system results

Table 4: Detection Rate

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Detection Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>98.5</td>
</tr>
<tr>
<td>Anomalous</td>
<td>98.82</td>
</tr>
<tr>
<td>Suspicious</td>
<td>98.85</td>
</tr>
</tbody>
</table>

Table 5: False positive Rate

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>False positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.029</td>
</tr>
<tr>
<td>Anomalous</td>
<td>0.03</td>
</tr>
<tr>
<td>Suspicious</td>
<td>0.02</td>
</tr>
</tbody>
</table>

b) Apache Spark Evaluation

To evaluate Apache Spark's structured streaming performance, there are many metrics [37][58].

In our work we will use just two important metrics, which are listed in Table 6 given below. For each metric a definition and a brief description are given.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Rate</td>
<td>Num Records/Input Time Sec</td>
<td>Describes how many rows were loaded per second between the start of the last trigger and the start of the current trigger</td>
</tr>
<tr>
<td>Processing Rate</td>
<td>Num Records/processing Time Sec</td>
<td>Describes how many rows were processed per second during the start of the current trigger and the end of the current trigger</td>
</tr>
</tbody>
</table>

When it was launched, Spark processed all files published from 2007 until June 2019, 613027 rows were collected after applying deduplication operation to remove duplicate records.
As shown in table 7, thanks to max Files Per Trigger, which set to max and can be limited to one or more files collected at a time, our proposed system was able to process 60635 records in one second and collected all files in one batch.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Rate</td>
<td>613027 Records/Second</td>
</tr>
<tr>
<td>Processing Rate</td>
<td>60635 Records/Second</td>
</tr>
</tbody>
</table>

The following days, our system automatically ingested a new file and predictions were made with the same performance.

In general, our system showed good classification results thanks to graph machine learning algorithm, and apache spark structured streaming.

VIII.  CONCLUSION AND FUTURE WORK

In this paper, we propose a real-time Intrusion Detection System based on the graph to detect and classify intrusions. Our proposed system is distributed and scalable due to the performance of Microsoft Azure. Our graphical machine learning algorithm k2 showed good results thanks to the Spark Machine Learning library which make the implementation of Machine Learning algorithms simple ans easy. Also, we use Spark Structured Streaming, which provides multiple and important functionalities to process intrusions in real-time. We used MAWILab dataset to evaluate our proposed system. Based on experimentation, our system achieves great results with good processing speed using only a small cluster.

In future work, we propose to use multiple clusters to achieve faster results and combine two or more datasets. In addition, we will develop our system with a deep learning algorithm such as neural networks to detect, classify anomalies, and get better results.

REFERENCES Références Referencias


33. Matthew Marden and Carl Olofson, the Business Value and TCO Advantage of Apache Hadoop in the Cloud with Microsoft Azure HDInsight, International Data Corporation (IDC), 2015.

34. Apache Spark Available online: https://spark.apache.org/docs/latest/ (accessed on February 10, 2019)


39. ML pipelines Available online: https://spark.apache.org/docs/latest/ml-pipeline.html


52. Apache Parquet vs. CSV Files Available online: https://dzone.com/articles/how-to-be-a-hero-with powerful-parquet-google-and (Accessed 12 October 2019).


56. C. Ruiz," Illustration of the K2 Algorithm for Learning Bayes Net Structures".

