ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse15942020.pdf
https://doi.org/10.30534/ijatcse/2020/15942020

Detecting Stealth-based Attacks in Large Campus Networks

Mohammad Al-Fawa'reh[1], Mustafa Al-Fayoumi[2]
[1]King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan, fawareh@outlook.com
[2]King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan, m.AlFayoumi@psut.edu.jo

ABSTRACT

Detecting and classifying new malicious network traffic is a high priority concern for cybersecurity practitioners. New stealth or zero-day attack can make companies go out of businesses in the digital transformation era. Despite the plethora of studies that have explored different machine-learning (ML) techniques to address this issue, the most popular used approach remains traditional ML with legacy datasets and small campus network. The difficulty in data collection considers the biggest impediment of using ML. This paper examines the possibility of exposing zero-day malicious network traffic in large campus networks based on cloud environments by presenting a lightweight framework. An experiment was devised for the analysis. However, before that, the characteristics of the network were examined based on the flow level. The framework showed an outperformed accuracy rate of 100% for a specific type of attack and 97.97% as a comprehensive detection mechanism.

Key words: AWS,Big Data, BoT, DoS, DDoS,Cloud computing, Cloud dataset,CSE-CI-UNB 2018, ML, PCA, WEB Attack, Zero-day attacks

1. INTRODUCTION

Innovative technology like IoT, cloud computing, big data, and data mining has generated much interest due to playing a vital role in creating a new business and converting the world to an infinite number of information and communication systems. However, using these innovative technologies exposes users to critical vulnerabilities, and can potentially be used inappropriately in cyberwar [1]. Since many companies store sensitive information in cloud servers, attackers can take advantage of some vulnerabilities and craft attacks to steal stored information. Moreover, IoT poses a broad range of cyber-risks that not only jeopardize businesses but also include banking, education, government, and health care systems. Statistics show the rise of cybercrime worldwide. According to [2], “The annual cost of cybercrime damage is estimated to cost the world $6 trillion by 2021. That is a major jump from $3 trillion in 2015, with cyber-attacks now one of the biggest threats to any business“. Cyber-attacks are becoming increasingly sophisticated, with cyber hackers developing new and determined threat methods that are becoming increasingly difficult to detect, making attacks more dangerous than ever. Traditionally, assets (systems, networks, process, etc.), have been protected by prevision systems. For example, the Web Application Firewall (WAF) has been used to filter malicious requests by checking URLs.

Based on a listing system (i.e. with blacklisting and whitelisting of individual URLs). Prevision systems include authentication authorization and accounting servers (AAA). However, these prevision systems are susceptible to be defeated using different techniques like fragmentation, imitation of legitimate users, and flooding. New stealth vulnerabilities are continuously discovered, and intruders keep developing attacker methods and mentality changes over time. IDS has arisen as a potential defensive mechanism, but existing systems suffer from many high false positive and negative reports, in addition to the fact that we cannot handle zero-day attacks. These manifest challenges underpin the motivation of this research and the approach developed to build a robust back-end engine that is faster in detection and construction compared with existing approaches, with minimal false reports (positive and negative).

In recent years, existing IDS has suffered from many issues, like the high rate of false-positive alerts. Too much time is taken to capture an attack, as well as the difficulty of detecting novel attacks. With the high false-positive rate, an attacker can build a simple tool using any programming language, like Python, to generate a bundle of false-positive alerts. Hypothetically speaking, IDS generates multiple alarms for the IT team. By default, the team will start digging to figure out if their company is under attack and if they should take action. The team may alternatively shut down the servers or isolate them from the Internet. In this time, the attacker would compromise a vital angle of CIA (availability). The attacker can also flood the system with false-positive requests, and when the IT team check those requests, this will create the perfect opportunity for an attacker to launch an attack attempt. Persistent research and development endeavours are being made to diminish the high false-positive rate of existing systems. It is well known that IDS is a process of data analysis and can be considered problematic in terms of correctly classifying data. From this point of view, it may also be observed that any classification process depends primarily on how the dataset is suited to the tools deployed. Using more real and clean data increases the potential for more precise outcomes. This implies that if the model can extract proper features that mark normal data from abnormal data, we can significantly reduce the false positive rate to an acceptable level.

We must also note that most IDS variants based on data mining and ML learning use well-known analytical techniques. These general techniques may not be instrumental in classifying data as usual or undesirable with a high degree of accuracy. Therefore, we need to alter and enhance existing systems to overcome the limitation of the detection mechanism. An attacker's techniques change over time. In addition to the problems mentioned above, an attacker can use advanced techniques like AI and ML to build a model to test the ability of the defense lines, like reinforcement learning. Briefly, the present paper attempts to answer the following research question:

How can we make IDS fast enough to process data on the cloud and detect stealth attacks using anomaly-based IDS, and how can we reduce the false-positives alerts to an acceptable level with a high detection rate?

This paper compares various algorithms with and without dimensionality reduction approaches to evaluate which ML algorithms have the best overall results when performing behavior analysis, based on the following criteria:

- Classification efficiency.
- Time consumption in the training phase only.

1.1 Machine Learning-Based Anomaly Detection

Anomaly detection technology works by learning machine, and by extracting normal activity by learning patterns from network traffic [3]. Many of these algorithms can support online learning and predict network flow. Thus, IDS identifiers that use these types of algorithms can adjust their learning parameters and possibly perform better than other anomaly detection techniques. However, this can be considered a drawback due to the large resource consumption. Another important drawback is the increase the overfitting [4] because some of these technologies are computationally complex and can learn the details or noise of network traffic received [5]. As a result, poor performance can appear in the classification of new incidents in the network traffic received. Depending on the characteristics of network traffic, many unsupervised and controlled ML methods have been introduced. For example, Markov chain diagrams that call for a sequence of activity from previous observations [6], Bayesian analysis of the relationship between features and forecasting of future events [7], and aggregation are also an effective way to perform external detection by separating the observed data based on the proximity scale [8]. Genetic algorithms and neural networks are inspired by biology [9]. For example, a neural network that simulates the operation of human brains usually uses ML methods to detect anomalies.

1.2 Network Behavior Analysis

Network Behavior Analysis (NBA) is a set of activities that improve system integrity by examining traffic and reviewing network information from existing infrastructure devices [10]. Network analyzer systems such as TCP dump (Daniel. 2019), network miner[11], and Wireshark [12] are used in a wide range of applications, from network activity logging to spyware detection and reverse engineering protocols. Nowadays, the NBA's premium technologies and tools are used as elements in many types of anomaly detection systems. NBA aggregate stats for entire traffic on a packet or flow level.

1.3 Packet & Flow Level Analysis

Packet level analysis focuses on examining individual packets obtained from network traffic in real-time. The "NBA" element checks for a very useful header and data payload after capturing packets. In conjunction with header data, some indirect characteristics can also be obtained by assessing the level of packets, including rate, arrival times, and hash rates. To assess the packet level, the packet sniffer is used while recording traffic, tracking network performance [13], identifying bottlenecks, collecting lost information, and discovering intrusions [12].

A network flow is a unidirectional sequence of IP packets that pass through nodes (router, switch, or even host) in a given time period [14]. Packages belonging to the same stream must contain the same header fields, such as port information, protocol type, destination address, and source addresses. The total number of flows summarizing the full communication builds the flow record. Reports indicate that users communicate with another party when this connection occurs, which is the transmission technology and other characteristics of a particular connection [15].

Flow analysis can also provide many indirect statistics about network information, similar to packet analysis. Flow logs can create different bulk inputs from all raw data. Using only statistical data (and not full IP packets) makes it possible to achieve relatively small flow records as opposed to raw production [15]. As a result, flow assessment is mostly performed on routers and the old firewall used with NIDS, where different hosts can be monitored by the detection tools. Argus [16], SiLK [17] and FlowScan [18] are well-known flow analysis programs.

The rest of the paper is divided as below. Section 2 includes the background and previous work. Sections 3 and 4 clarify the methodology for the research with the implementation. Section 5 addresses the outcomes. Finally, the paper concludes with Section 6.

2 RELATED WORK

Network behavior analysis have attracted considerable attention to the research community and the number of publications is increasing from year to year. In this section, we concentrate on work related to anomaly detection due to the limited space. Researchers classified IDS based on the working mechanism to Signature-based and Anomaly-based. The Anomaly classified into Adaptive Learning (or Self-Learning) and Manual Learning (Programmed Learning). The programmed model is when the system needs a participant or external person to teach the system how to identify behavioral changes. The participant decides the extent of the abnormal behavior of the system and knows the possibility of penetration [19]. There are three types of programmed models: basic rule-based, threshold, and numerical models. In this paper, we will focus only on adaptive methods, the self-learning system works by example in a basic sequence. While self-learning is done by creating a model for observed machine traffic that has accumulated over a period of time for basic operations [20]. Self-learning models are divided into the following main categories: machine, deep learning, and time series model.

2.1 Time Series

The time series model takes the observation sequence into account for succession occurring at standardization periods. If the possibility of a new event at some point is trivial, then a shift in normal behavior can be considered. The time series has the advantage of monitoring behavior trends and distinguishing them over a period of time if you notice a change in normal behavior. An example of a time series model used as identifiers is ARMA acronym for Automatic Reflection Moving Average. If attacks continue over a period of time [19], it is an active template. However, this model has the disadvantage of being more computationally expensive [5]. GARMA and ARMA series models that are configured to identify 4 types of attacks (probe, DoS, L2R, and U2R) [5]. Parameter calculation was performed using the Hannan-Rissanen equation, Whittle estimate, and probability maximum approximation, and the expected point resulting from Whittle estimate and maximum probability was close to the actual value. Time series models were able to predict the attack, but GARMA's performance in detecting the attack was better.

2.2 Machine Learning Approaches

ML approaches are systems for detecting distortions independent of humans. They determine anomalies over a period of time by revealing the irregular features in a system [21]. The effectiveness of this technique is its ability to distinguish within the network between normal and anomalous situations. Without comprehensive human training or intervention, ML can provide IDS methods for detecting existing, new, and light attacks. It is described as a set of methods that can detect patterns automatically to predict future trends in data [22][23]. This section discusses ML and DL known in the field of anomaly detection.

A. Supervised learning

Barapatre et al. [24] experimented with an input layer, a hidden layer, and an output layer in the neural network. The input node contained 41 features of the KDD Cup'99 dataset. The output node in the dataset was identified as normal or attack. 0 means normal data, and 1 means attacks. First, the learning rate (LR) was set to "0.1" and the program was retrained to lower levels of learning. The sigmoid activation method was also used. The program is trained to assess the quality of the algorithm on individual attacks for each class of attack. The researchers concluded that MLP-BP NN was more effective in detecting DoS and Probe attacks than U2R attacks. They also reported an increase in the identification rate as the LR decreased, resulting in a gradual affinity. They also noticed a decrease in the LR and the RMS value (SSE) and the retrained network showed an increase in detection rate, but a small and shallow model was used, while the current study uses 16 million records. The work presented in [25] used a special type of anterior feeding neural network called the Radial Basis (RBF) function and compared the performance of RBF and MLP-BP in identifying four different types of attacks on the KDD Cup'99 dataset. Training and test samples included 1,000 records from the selected dataset. All samples were different in that experiment with 34 numerical properties and 7 symbolic properties. Before being used as training or test information, symbolic features were encoded in ASCII numbers. The results showed that RBF performed better than MLP-BP with a 99.2% detection rate and a 1.2% false-positive rate. An early test method using the ISCXIDS 2012 dataset was proposed by [26], which included classifiers for the K and Naïve Bayes means. By selecting the incoming packets of a different host in a single day, the proposed algorithm detection was studied, but using K-mean and Naïve Bayes is slow to categorize datasets with many advantages. [27] Implemented a multi-target genetic algorithm for both the 1999 KDD Group and the 2012 ISCX-IDS subgroup. Decision Trees (DT) were developed by [28] based on Snort IDS alerts. In this analysis, only five features were extracted from the dataset (protocol, source and destination IP, source and destination port), and decision groups were created using these attributes. [29] developed the ISCX-IDS 2012 dataset flow level and tested it for other group learning. In converting the package to flow, the Flowcalc tool was chosen, and relatively basic statistics were extracted from the dataset [30]. Tan et al. [31] analyzed traffic data by converting traffic to images. This paper focused on reducing DoS attacks and applying a special distance meter called Earth Mover's Distance (EMD) to the principle of object discovery. In addition, PCA has been applied to the intended network traffic, containing only basic features. Likewise, by applying different ML algorithms to the ISCX-IDS 2012 dataset, [32] studied the effect of PCA on intrusion detection.

B. Unsupervised learning

NN is provided only with input data in unattended learning without visualizing the outcome. It independently finds trends in the data. Undiscovered data are referred to as unnamed data [33]. Adaptive resonance theory (ART) and self-organizing maps (SOMs) are common.

- Adaptive Resonance Theory

Murphy et.al proposed a hybrid method for categorizing different attacks on the 99 KDD Cup dataset using the PCA and Fuzzy counter-resonance theory [34]. The results showed a high detection rate of 96.13 and a false alarm rate of 3.86 for the detection of anomaly violations. While Chauhan et al. [35] implemented a new technique by loading the network weights based on the score of the evaluation function [36]. Using different case possibilities, an effective evaluation function was chosen. It follows that if the weights in the ART network fluctuate, they are broken down, and thus ART can detect any intrusions.

- Self-Organization Maps (OM)

SOMs are usually used as a mechanism to detect anomalies, which can be used as a host-based detection system for snooping. [37] Studied and demonstrated two basic feature sets with the SOM hierarchical design. One of the primary features with all 41 features is limited to 6. Results showed a 90.4 detection rate and a false positive rate of 1.38. While [38] presented a single SOM architecture method that discovered attacks using various parameters on MANET. Its experimental results in terms of detection rate and false alarm rate were found to be higher than other neural network approaches.

The limited number of researchers used CSE-CICIDS2018 Dataset, and most of them used the unsystematic approach during the preprocessing, their works summarized in Table 1.

Table 1: Related Studies of CSE-CI-UNB 2018 Dataset

Literature	Attack types	Classification Algorithms	Accuracy Rate
Kanimozhi & Prem Jacob, 2019	Bot	ANN	99.97%
Kanimozhi, 2019	BoT	1- K-Nearest Neighbors	99.73%
		2- SVM	99.98%
		3- Decision tree	99.9%
		4- Random Forest	99.83%
		5- Naïve BAYES	99.92%
		6- Neural Network	99.97%
Qianru Zhou,2019	All Attacks	1-Random forest 2-Gaussian naive Bayes 3-Decision tree 4-Multi-layer Perceptron 5-K-nearest neighbors 6-Quadratic discriminant analysis	96%

3 CSE-CI-UNB 2018 DATASET

Many of reference datasets can be accessed to assess intrusion detection techniques and systems. Various attack scenarios were produced in these datasets using simulation environments. The KDDCup99 [39] dataset is one of the oldest and most well-known IDS datasets. It is collected within a seven-week period of tagged information that includes 41 properties. The training dataset consists of approximately 4,900,000 unconnected vectors with 24 differentiated attacks, while the test dataset includes 300,000 samples with 14 additional attacks [40].

There are four types of attacks in the dataset: user to root (privilege escalation), remote to local (buffer overflow), denial of service, and scanning. Because this dataset has many issues that lead to poor evaluation of anomaly detection, an NSL-KDD [41] dataset has been introduced to address these problems. The NSL-KDD dataset contains only specific records from the complete KDDCup99 training dataset. Additionally, the NSL-KDD dataset does not contain duplicate or redundant records. As a result, the ML algorithms used during intrusion detection are unbiased toward the most common records [42]. The training and test suite includes multiple attack samples at a significant percentage. This leads to a more accurate and similar performance evaluation, as there is no need to choose a portion of the dataset to assess the efficiency of premium penetration detection systems. [43] Introduced the new DARPA intrusion detection dataset. This dataset was developed by MIT Labs to detect complex attacks that have different stages. Attack cases were simulated through test sessions, infiltration into the system by exploitable vulnerabilities, as well as the installation and launch of DDoS attacks on other hosts [31].

Another unrealistic dataset intruded by DEFCON [44], which was collected in the hacking contest, which only contained a spam action. This dataset is out of network traffic in the real world and therefore, has restricted usage. Finally, the [45] dataset is a collection of various types of network information that can be accessed for study purposes. However, this dataset is limited to specific incidents and interventions, such as DDoS. Although reference datasets are useful in assessing intrusion detection systems, many of them have problems such as unrealistic network setup, unmarked or incomplete information, restricted intrusion scenarios, and the excessive ratio between regular traffic and attacks. Realistic datasets have been produced in recent years to create realistic network scenarios. For example, a UNIBS data collection [46] was collected from the University of Brescia. The dataset was developed for three consecutive days by the TCP Dump program running on the college router and storing it with 20 devices. [42] Used the TUIDS dataset, prepared by a scientist from Tizpur University. This fully featured dataset is recorded at the flow and packet level. An ISCX-UNB dataset [28] was created by creating worker profiles, and multi-stage attack cases were produced to produce intrusions. Six years later, the same research center, in integration with Amazon (AWS), created a more realistic and updated suite with more attacks, such as web attacks included. The method proposed in this paper was evaluated using the CSE-CIC-IDS 2018 dataset. This dataset consists of two realistic profiles - one to mimic normal activity and the other for abnormal activity. Table 2 shows the statistic for every attack and the number of normal and malicious flows, whereas Tables 3 and 4 compare this dataset with the available public datasets.

Most of the research in anomaly detection or prediction uses public datasets like PREDICT, CAIDA DEFCON, ADFA, NSL-KDD KYOTO, and ICS Attack. Every one of these datasets has pros and cons, but we will not discuss them in detail because they are not part of our research. However, that dataset is rare and extremely legacy for many reasons, which are summarized below:

- Behaviors of the adversary change over time.
- Most of the datasets did not consider the web attacks on threat modeling since the majority of businesses rely on the web.
- A few studies investigate malicious network activity in the cloud by using large campus networks and large datasets.

All these reasons encourage us to use a more realistic dataset, so we proposed using a public and realistic dataset (to reduce the gap between our study and other research studies) from Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC), hereafter referred to as CSE-CIC-IDS-2018. The dataset includes seven attacks that use different techniques through several scenarios: web attacks, distributed denial-of-service (DDoS), brute-force, and infiltration of the network from inside, botnet, Heartbleed, and finally, denial-of-service (DoS) attacks. The design of the network and intrastation also simulates the original design found in the industry since the number of machines is 500, which is considered a relatively large number so that the machine is distributed as below in Figure 1.

Table 2: Comparing Different Datasets

Data	**Realistic testbed Setup**	**New Attacks**	**Labeling**	**Features Extraction**	**Mirroring**
DARP	T	F	T	F	T
KDD99	T	F	T	T	T
DEFCON	F	F	F	F	T
LBNL	T	F	F	F	F
CDX	F	F	F	F	T
KYOTO	T	F	T	T	T
TWENTE	T	F	T	F	T
UMASS	T	F	T	F	T
ISCX2012	T	F	T	F	T
ADFA2013	T	F	T	F	T
CICD2018	T	T	T	T	T

Table 3: Comparing Different Datasets

Data	**HTTP**	**HTTPS**	**SSH**	**FTP**	**SMTP**
DARPA	T	F	T	T	T
KDD99	T	F	T	T	T
DEFCON	T	F	T	F	F
LBNL	T	F	T	F	F
CDX	T	F	T	T	T
KYOTO	T	T	T	T	T
TWENTE	T	F	T	T	F
UMASS	T	F	F	F	F
ISCX2012	T	F	T	T	T
ADFA2013	T	F	T	T	T
CICD2018	T	T	T	T	T
CICD2018	T	T	T	T	T

4 METHODOLOGY

This section presents the research methodology in detail and the main pipeline is summarized below.

- Data Collection
- Feature extraction and removing unrealistic features
- Data pre-processing
- Splitting data in training and testing
- Model building and evaluation
- Using PCA with ML Models
- Combining all data together and validating it with and without PCA

Table 4: Summary of CSE-CIC-IDS-2018 Dataset.

Day (2018)	**Normal flow**	**Attack**	**Attack description**	**Day (2018)**	**Normal flow**	**Attack**	**Attack description**
14-02	667626	193360	FTP-Brute Force	22-02	360833	249	Brute Force-Web
		187589	SSH-Brute force			79	Brute Force-XSS
						34	SQL Injection
15-02	996077	41508	DoS using GoldenEye	23-02	1048213	362 151	Brute Force-Web Brute Force –XSS
		10990 139890	DoS using Slowloris Slow HTTP Test			53	SQL Injection
						362	Brute Force-Web
16-02	446772	461912	DDoS using Hulk	28-02	544200	68871	Infiltration
		576191	DDoS using -LOIC-HTTP				
20-2	7372557	1730	DDoS using -LOIC-UDP	01-03	238037	93063	Infiltration
		686012 193360	DDoS using -HOIC FTP-Brute Force				
21-02	7372557	187589	SSH-Brute force	02-03	762384	286191	Bot
		41508	DoS using GoldenEye				

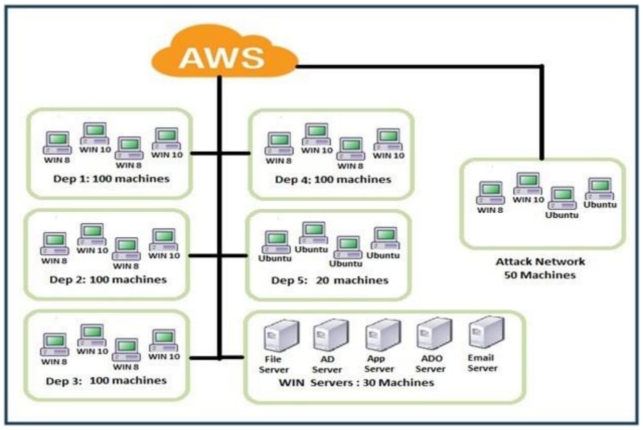

Figure 1: Network Architecture of the Dataset.

4.1 Data Collection and feature extraction

The dataset was collected from CSE-CIC-IDS-2018 as we mentioned in section four. In order to extract the features from the raw PCAP files based on flow level, we used statistical approaches through CICFlowMeter-V3 tools and we obtained 84 features organized in Table 5.

4.2 Data Preprocessing

This is the most important step in data mining and the accuracy of the model depends on the data preprocessing, the more we introduce clean data, the higher the precise outcome we will get. Data preprocessing include three main steps: removing duplicate values, remove missing values, and implementing data standardization. Removing duplicate values Record redundancy is the main problem on various datasets, due to the fact that the duplication will consume the resources and will increase the time of the training especially when we are dealing with the large dataset so to overcome this issue we will use ’remove duplicate method’ using Kutools for Excel and Panda library. Remove Missing and unknown values missing and unknown values like null. Values will affect the performance and accuracy, so dealing with those values is a mandatory step, In fact, there are three well-known methods to dealing with missing and unknown values. Complete Case analysis (CCA) by removing the whole line if one or more than one feature is missing or invalid values such as Nan and Infinity; replacing the missing value by the median for those columns; and replacing the missing or unknown values by zero. In this paper, we used the CCA method due to the many bad requests and replies to the PCAP files from one hand. Moreover, this method is faster than other methods.

4.3 Normalization

It helps us to normalize the data within a particular range. In addition, it also helps in speeding up the calculations in an algorithm, since we are dealing with continuous features, the following algorithms will affect the accuracy:

- K-Means uses the Euclidean distance measure; feature scaling matters here.
- K-Nearest-Neighbors also require feature scaling.
- Principal Component analysis (PCA): Tries to get the feature with the maximum variance; feature scaling is also required here.
- Gradient Descent: Calculation speed increase as Theta calculation becomes faster after feature scaling.

In this phase, we used the standardization scaler as shown in the equation: $z=(x-u)/s$

In order to populate our IDS-engine, we removed some features: Flow ID; Protocol; Source IP; Destination IP; Timestamps; and finally Source Port Number, due to high data cardinality. These features differ from one network campus to another. In the second part of the experiment using PCA, we only used the Dst Port number and Flow duration. To the best of our knowledge, there is no relevant research that used this dataset containing 76 features.

4.4 Proposed Approach

This section briefly summarizes ML techniques we used during this research.

Table 5: Features Description.

Features	**Description**
Flow ID	Unique identifier for flow record
Src/ Dst IP/ Src /Dst Pt	Source/Destination IP address, Source / Destination Port #
Pt/ fl dur /TP	Protocol/ Flow Duration / Timestamps
Tot fw pk/tot bw pk	Total # of packets In forward/backward
Tot l fw/Bw pkt/max/min/avg/std	Total Max/Min/Avg/Standard deviation Packet size in Forward/ backward Direction
Fl iat avg/std/max/min	Average/Standard deviation /Maximum/Minimum Time between two flows
Fw/Bw iat tot/ avg/std/max	Total/Mean /Maximum /Minimum /Standard deviation of time between that sent in Forward/Backward direction
Fw/bw psh/urg flag	# of times that URG/Push are setting to 1 in forward/backward direction
Pst/ack/urg/cwe/ece / syn/rst/fin cnt	# of packets with PUSH/FIN/SYN/RST/ACK/CWE/ECE/Urg Flags
Pkt len min /max/avg/std	Mean/Minimum /Maximum/Standard deviation of that length flow
Fw/bw pkt s	# of packets in forward/backward per sec
Fw/bw byt/pkt blk avg	The Average # of packets/bytes bulk rate in forward/backward direction
Subfl fw/bw pk/byt	The average # of packets/bytes for sub flow in forward/backward direction
Atv min/max/std/avg	Minimum /Maximum/Standard deviation/ average time a flow was in active mode before switching to idle state
Idl min/max/std/avg	Minimum /Maximum/Standard deviation/ average time a flow was in idle mode before switching to active
Up/Download ratio	upload and Download ratio
Pkt size avg	the average of packet size
Fw/bw seg avg/min	the average/Minimum of packet size in Forward/Backward direction
Fl byt/pkt s	# of packets /bytes are transferring per sec
Fw/bw hdr len	Total bytes in the header in forward/backward direction
Fw/bw win byt	# of bytes sent in forward/backward direction for initial window
Fw act pkt	# of packets with at least 1 byte of TCP data payload in the forward direction
Pkt len va	Minimum inter-arrival time of packet

A. K-Nearest Neighbor (KNN)

KNN is used in supervised learning classification systems requiring prior sampled data distribution [47]. Sample xi is identified by testing the closest neighbors. A set of sampled data x1, x2... xn, is transformed into a metric space when constructing the closest neighbor rule. The space metric is designed with a measure of similarities, such as distance functions of Manhattan or Euclidean. This method is used to obtain a new query point xi nearest neighbors. Finally, the closest neighbors are selected to determine the class by majority votes [47]. While KNN performs very well, as it searches for the base separation in all the preparation set for one test, it has high time complexity. In this preliminary work, the label of a request point was predicted by five equally weighted closest neighbors, and the similarities measure was chosen as the Manhattan and Euclidean distance. The Euclidean method takes the difference between the coordinate, and after that takes the square root of the sum of the difference, for example, if m = (f,g) and w = (t,r), the Euclidean distance between these two points is given in Equation 1 [48].

$$\sqrt{(f-t})^{\wedge}2 + (g-r)^{\wedge}2 \qquad (1)$$

While Manhattan Method takes the difference between the coordinate and then take the sum of the absolute value as shown in Equation 2 [49].

$$|f-r| + |g-r| \qquad (2)$$

B. Decision Trees

Decision trees (DT) are nonparametric-based ML methods that can be used for classification and regression tasks. The decision tree building process is done in the binary division, as shown in Figure 2. The final nodes in the decision tree lead to different class ratios in a particular region. Therefore, a suitable testing area has been established and it is expected that all test data will be the most frequent category in the region [28]. The Purity of a node assesses the value of the divisions in the DT. The Purity of trees can be determined using several methods, known Entropy, and Gini. In Equation 3, cross-entropy is introduced [50]. In the equation, Pmk shows the percentage of training samples in area m of group k. If Pmk is all near-zero or close to one, a lower value is provided by Entropy [51]. Therefore, for this node, the more negative Entropy node produces more samples that are dominated by one group and have the lowest classification error).

$$Entropy = -1 * \sum_{1}^{k} Pmk * \log(Pmk) \qquad (3)$$

DTs are easy to interpret, and require minimal data preparation and computational complexity to reduce logging time. However, as they are affected by slight differences in training samples, they appear to be unstable [52].

$$\text{Gini} = \sum_{1}^{k} P(i) * (1 - p(i)) \qquad (4)$$

Gini impurity approach addresses the probability of misclassifying a randomly selected item in the dataset randomly labelled according to the classification distribution in the dataset. It has given in Equation 4 [50].

Where K represents the number of classes and p(i) represents the probability that an element of class I is randomly selected. Through maximizing the Gini Profit, which is determined through extracting the measured impurities of the branches from the original impurity, the best split is selected when training a decision tree.

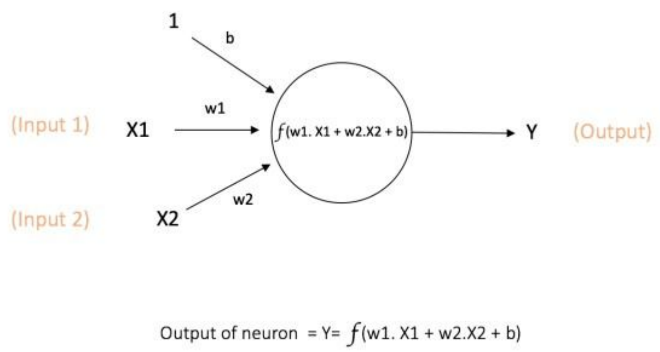

Figure 2: CSE-CI-UNB 2018 Dataset

C. Random Forest

Random forest (RF) [54], considered a well-known ensemble learn algorithms, is an algorithm used by Microsoft for games. It is inspired by the bagging of the decision tree. Bagging is a general technique aimed at reducing learning process variance [52]. As mentioned in the previous section, DTs have a great variance; therefore, bagging can improve overall quality. A portion of the data is selected and the DT is prepared for the sample in the RF classification training phase. This procedure is repeated B times. New samples are classified by a majority vote in the classification phase [52]. RFs are a robust, highly accurate form of ML. In fact, even for larger datasets, they can achieve good performance [9]. The effects of overfitting are uncommon in RFs because they are less susceptible to changes in inputs [55].

D. Perceptron

Perceptron classifier measures the linear mixture of input characteristics and detects a separate hyperplane [9], [51]. It gives the position of the sample question regarding the separator plane. Although Perceptron is a simple and simple classifier used for minor classification problems, it is included and evaluated in the dataset because it is a modern linear classification method on the latest model and can be contrasted with other linear models in this part. It is also known to be the smallest element in the neural network [51] and ELM systems. Figure 3 depicts a simple perceptron model. As shown in the figure, a calculated amount of input is taken from sensory perception. This usually applies to this weighted non-linear activation function, expecting the input sample category with a predetermined threshold value [52]. The perceptron learning algorithm uses weights w0, w1, and w2, and w3 ..., wn to fit a different hyperplane into a decision boundary. The main objective of the method is to use different gradient algorithm like (SGD) to minimize the gap of misclassified samples [56]. The perceptron algorithm can be paired with one system for everything in multiple classes. While the perception is very fast even with big data, when the data is not linearly separate and distinct, the algorithm will not converge well. In addition, the solution depends heavily on the initial weight values [56].

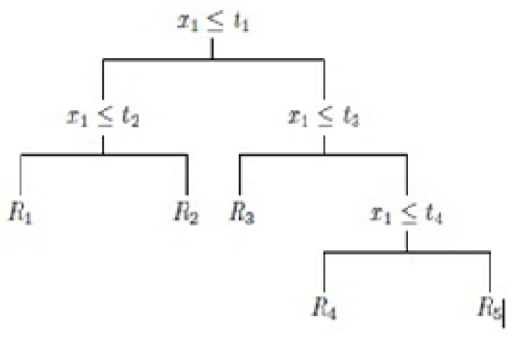

Figure 3: DT process [53].

E. Naïve Bayes

The first presentation of Bayes Theorem in 17011761, followed by a researcher named Pierre Laplace. Bayes Theorem explains the probability linked to the event based on previous knowledge of conditions [57]. Therefore, it is great for ML, because that is exactly what ML does (predicting future events on the basis of previous experience). Work is possible as a conditional model P(y1....Ni) for the data sample I with N outputs [57]. This model is described in Equation 5 by Bayes' theory [53].

$$P(\psi\kappa|\varkappa) = \frac{P(\psi\kappa) * P(\varkappa|\psi\kappa)}{P(\varkappa)} \qquad (5)$$

Where ψκ is the label of class κ. If it is possible to estimate priors P(ϰκ) and probabilities P(ϰ|ψκ) for the training data, the latter likelihood is extracted from the theorem of Bayes. Naïve Bayes model also capable of classifying the test data based on posterior probability, as shown in Equation

Choose ψi if $P(\psi i|\varkappa) = MaxP(\psi\kappa|\varkappa)$

In Gaussian Naïve Bayes, the data vector components are expected to be distributed in Gaussian, and statistically independent of each other for the provided category tag. Based on this assumption, the underlying distribution of the data is based on the Gaussian multivariate distribution N (), where are the location vector and covariance matrix. Hence the standard multivariate density of the previous distribution class k = 1, 2 ..., N of the observable samples can be extracted as:

$$\frac{1}{\sqrt{|\Sigma\kappa|(2*\pi)^d}} * e^{\left(\frac{-1}{2(\varkappa-\mu)}\right)^T} * \Sigma_\kappa^{-1}(\varkappa - \mu_\kappa) \qquad (6)$$

The classifier Gaussian Naïve Bayes is highly scalable, easy to implement, and can be applied to N-dimensional data vectors with low computational costs. Although it implies a provided category tag, a data sample is conditionally independent, and this approach can generate reasonably good results even if the data vector components are not distributed in Gaussian or the data is not statistically based [57].

F. Quadratic Discriminant Analysis

The last model we will use is the Quadratic Discriminant Analysis (QDA), a supervised ML classifier, and quadratic decision-making is similar to the Gaussian Naïve Bayes method. QDA estimates the posterior likelihood of P (1.......N). QDA also considers that samples observed have a standard multivariate distribution characteristic. If the data are slightly nonlinear [57], QDA can produce better results, as it assumes a quadratic boundary of a decision [58]. In addition, with larger training sets, it performs better as the effect of variance becomes less and less important for this process. Unlike Gaussian Naïve Bayes, QDA does not accept the concept of conditional independence for a class and notes that the separation mark does not allow data vectors to be independent of other observations. Therefore, if the contrast matrix is diagonal, then QDA becomes the same as Gaussian Naïve Bayes. In the case of multiple groups, QDA determines the test sample mark by calculating the sum of subsequent allocations corresponding to Equation 6.

5 EXPERIMENTS AND MODEL EVALUATION

In this research, we are proposing to build a Hybrid model using Statistical approaches, PCA, and set of Algorithms like (DT, KNN, RF, ANN) and implementing this with different scenarios. This section briefly summarizes ML techniques with PCA and dimension reduction techniques applied to the CSE-CIC-IDS 2018 dataset. To verify the reliability of the dataset, these approaches are applied and evaluated. In a 64-bit Windows 10 computer with 16 GB RAM and 2.60 GHz CPU, the ML was implemented using Python 3.7.3, Numpy 1.16.2, Scipy 1.2.1, and SPYDER 3.3.3. Every approach was applied to 78 features, followed by two principal components of normalized data capturing 95% of the original data variance. Two techniques were used to assess overall performance: training testing (80%, 20%) respectively, and the cross-validation method. Metrics can be defined as follows in the context of network intrusion detection [9], [59].

Recall/True Positive Rate (TPR) = TP/(TP + TN) (7)

Accuracy is the ratio of the total number of valid predictions made by TP + TN to all predictions made by TP + FP + FN + TN.

Accuracy = (TP + TN)/(TP + TN + FP + FP) (8)

F1 score represents the harmonic mean of the recall and the precision. F1= (2*Recall* Precision))/((Recall + Precision))) (8) Training Time Evaluation The tests are performed using system time on the computer used to train these models. This can be done easily by setting the start and end time with the Python process time command, then taking the end time minus the start time (Table 6). This gives the time the process to go through. Therefore, you can see a command or parameter set to the start and end times of each training command at the beginning and at the end. Table 6: Time taken during training.

Table 6: Time taken during training

```
Start__time = time.time ()

classifier.fit (X__train, y__train)

end__time=time.time ()

time.taken = end__time – start__time
```

Experiments in this paper were conducted in two phases: the first phase to apply the proposed framework separately to 10 different days, and the second by combining the flows generated from 9 days (not 10, due to hardware limitations) as a single large file.

ML techniques were applied to the dataset every day, and evaluation metrics were extracted for discussion. Figure 4 and Figure 5 provide a comparison of these techniques. All data samples were used in training on February 14. Hence, that attack was carried out during differentiated ports, and the accuracy of normal detection and attack data for many machine-learning algorithms achieved 100% from an accuracy point of view. In addition, from a time complexity view, Perceptron algorithms were the fastest in training. On February 15 all datasets were used in the training phase, as all of the algorithms achieved 100% accuracy, except the GNB, which achieved only 85% achieved, but the GNB was the best algorithm in terms of processing time, with 9.5 seconds, followed by Perceptron, needing 20 seconds to complete the training.

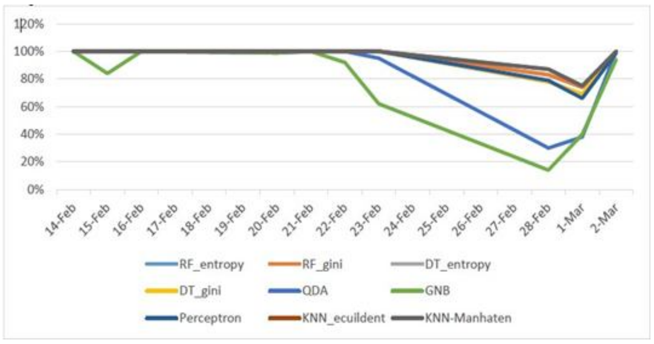

Figure 4: Accuracy comparison of various traditional ML without PCA.

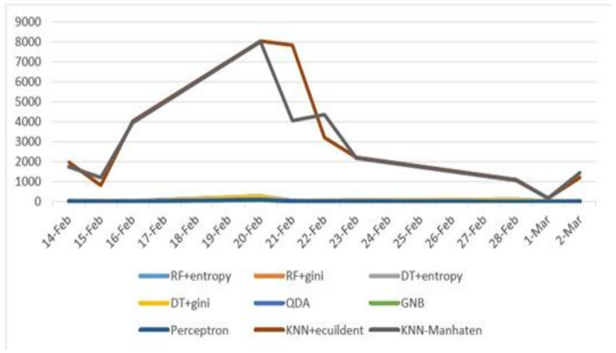

Figure 5: Time comparison of various traditional ML without PCA.

On February 16, all algorithms achieved 100% accuracy, and the learning algorithms achieved 100% from an accuracy point of view. Moreover, from the time complexity view, Perceptron algorithms were the fastest algorithm in the training need only 10 seconds. KNN was the worst algorithm in time processing (Figure 6, 7).

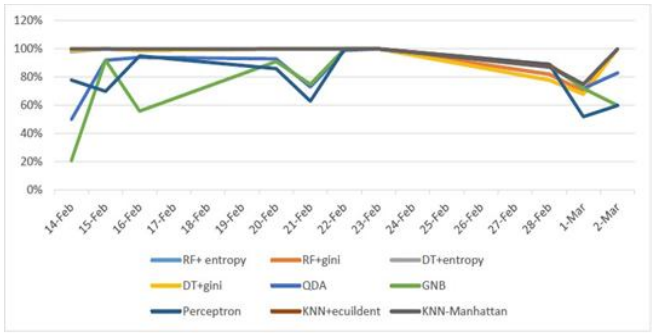

Figure 6: Accuracy comparison of various traditional ML with PCA (n = 2).

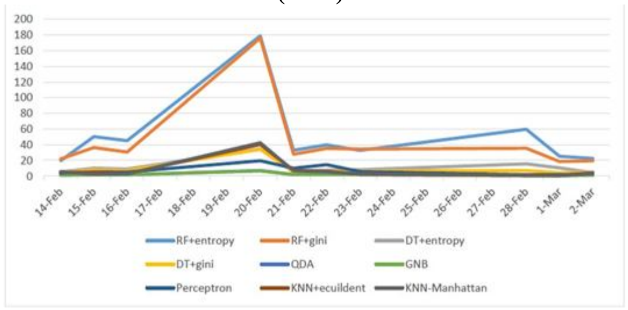

Figure 7: Time comparison of various traditional ML with PCA (n = 2).

On February 20, all algorithms achieved 100% accuracy, and Perceptron was the fastest algorithm with 14.75 seconds. On February 21, we got the same behaviors on February 15. All of them achieved 100% except GNB, with 92%, but the latter was the fastest algorithm in processing (15.91 seconds) On February 21, GNB was the worst algorithms in the detection phase with 62%, followed by QDA with 95%, but other algorithms achieved 100%. On February 28 and March 1, the worst performance ever seen in the training average of the detection was 69 and 65%, respectively. This discrepancy is expected due to infiltration attacks on February 28 and March 1. This type of hidden infiltration is difficult to detect because it appears to be legitimate flows, following protocol rules and handshake procedures (Garg Pollett, 2017). On the last day of the training, the performance increased and reached 100% on all algorithms except GNB (94%) and Perceptron (99%). Of these nine methods of ML, KNN and RF show better performance than other technologies. However, KNN and RFare complex and take a lot of time. KNN suffers from a dimensional curse [12], which leads to a massive increase in the size of training data. Therefore, KNN requires more calculations if a dimension reduction technique is not applied to data for simpler representation. However, Perceptron’s algorithms are better than other algorithms in terms of accuracy and time complexity. Consequently, the performance of the proposed method will mainly be compared with DTs, RF, and KNN. ML techniques using PCA were applied to the dataset every day, and evaluation metrics were extracted for discussion. Figures 6, 7, 8 and 9 provide a comparison of these techniques using PCA.

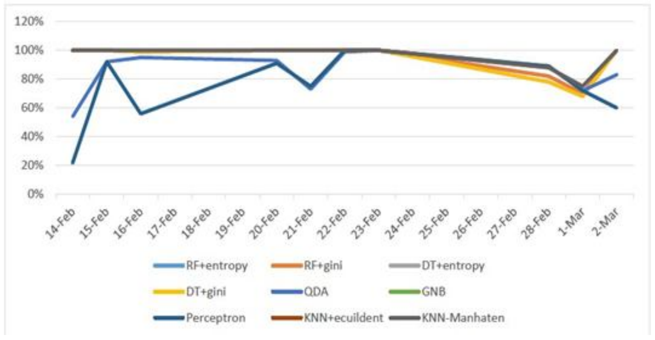

Figure 8: Accuracy comparison of various traditional ML with PCA (n = 2), using CV.

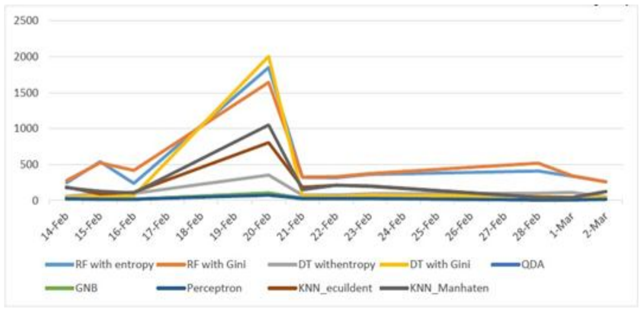

Figure 9: Time comparison of various traditional ML with PCA (n = 2) using CV.

After combining PCA with traditional ML approaches, a tremendous enhancement occurred in the training phase from a time complexity point of view. On February 14, the accuracy for KNN, and RF with Gini stood still at 100% and decreased slightly in performance by 1% for RF with Entropy, 2% for DT with Gini, 50% for QDA, 79% for GNB, and finally 22% for Perceptron. In addition, the time complexity for KNN decreased by 366 times. On February 15, the performance increased for GNB by 8%. However, it decreased only for two algorithms: Perceptron 30%, and QDA 8%. Moreover, the time complexity reduced by 525 times. On February 21, the time decreased for all models, however, performance decreased for GNB, QDA, and Perceptron. On February 22, the Performance increased for the GNB Algorithm, and the Accuracy reached 100%. However, QDA performance decreased by 1%. In addition, the time complexity decreased by 1920 times for KNN-Manhattan algorithm. On February 23, the performance peaked at 100%for all algorithms, with minimal time complexity. The performance also increased for all algorithms on February 28. On March 1, the performance increased in QDA, GNB, and Perceptron. However, the performance decreased on RF with Entropy, RF with Gini, DT with Gini, and DT with Entropy. While there is no in the accurate change attributable to KNN, the time complexity reduced by 230 for KNN, with Euclidean distance and 218 times for KNN with Manhattan.

On March 2, the performance is still 100% for KNN, RF, and DT. However, the performance decreased by 17%, 36%, and 40% for QDA, GNB, and Perceptron, respectively. The algorithm that used the least amount of time was DT Gini and PCA; as can be seen from Figure 8, it used less than one and a half minutes. The KNN classifier used the entire dataset over three days to process. This is the time it takes for the model to learn. As can be seen from Figures 5, 7, and 8, there is a major difference between the amounts of time used by the algorithms, both of which are controlled by the script and other processing factors, and changes in all scripts should be the same. In the second part of the experiments, we tested our hybrid framework on a large dataset, by combing all datasets except day 4 using the same methodology in the first experiments.

Table 7 shows that RF with Entropy archives the highest accuracy on binary classification, compared with a decision tree, RF, and others. We can see using PCA (n = 2) the accuracy will decrees by .05 and the training time to the Half. In addition, all algorithms achieved 97precision, recall, and F1 respectively. The results showed the robustness and lightweight our model in a large dataset with a minimal number of features (destination port and flow duration).

Table 7: Result of binary classification.

Algorithms	Time (Sec)	Accuracy	P	R	F1
RF and Entropy	1436.51	97.362%	97%	97%	97%
RF with Entropy and PCA	841.55	97.104%	97%	98%	97%
RF with Gini	1392.34	97.358%	97%	97%	97%
RF with Gini and PCA	724.22	96.918%	97%	97%	97%
DT with Entropy	1518.72	97.362%	97%	97%	97%
DT with Entropy and PCA	271.58	97.104%	97%	98%	97%
DT with Gini	1505.62	97.358%	97%	97%	97%
KNN	259200	97.47%	97%	97%	97%
DT with Gini and PCA	162.49	96.918%	97%	97%	97%
KNN and PCA	142.06	97.36%	97%	97%	97%

Table 8 shows that RF and Entropy model achieves the highest accuracy overall algorithms when multiclassification is used. The performance of the proposed model with and without dimensionality reduction was evaluated. Additionally, the outcomes with different ML techniques for detecting malicious activity, which were previously mentioned in Table1, were compared. Afterward, the algorithms were tested based on the accuracy of all the attacks. From the experiments conducted, our hybrid model was found to outperform the other algorithms in Table 1 as well as the proposed model by [60], [61] and [62].

Despite this paper's systematic approach, the results should be interpreted with caution due to some limitations relating to both the experiment and the dataset we used. [63] The main limitation of this experiment was the size of the dataset. In particular, the poor false-positive rate of some classifiers can cause a relatively small number of web attacks. Furthermore, the initial dataset might not have been sufficient for the qualitative learning process of the model. The second limitation regards the labeling of a training dataset. Because the data labeling of malicious traffic was done manually by authors, the quality of the labeling is not fully reliable. The third limitation is the hardware limitation of handling the whole dataset. In the last experiments, when we combined all the data into a single file, we dropped the data from 20-02-2018.

Table 8: Results of multi classification.

Algorithms	Time (Sec)	Accuracy	P	R	F1
RF and Entropy	1436.51	97.362%	97%	97%	97%
RF with Entropy and PCA	841.55	97.104%	97%	98%	97%
RF with Gini	1392.34	97.358%	97%	97%	97%
RF with Gini and PCA	724.22	96.918%	97%	97%	97%
DT with Entropy	1518.72	97.362%	97%	97%	97%
DT with Entropy and PCA	271.58	97.104%	97%	98%	97%
DT with Gini	1505.62	97.358%	97%	97%	97%
KNN	259200	97.47%	97%	97%	97%
DT with Gini and PCA	162.49	96.918%	97%	97%	97%
KNN and PCA	142.06	97.36%	97%	97%	97%

6. CONCLUSION

In this paper, we have presented a lightweight framework for zero-day attack detection based on cloud-dataset by applying ML techniques with a minimal number of PCAs. Our research aimed at demonstrating the potential and effectiveness of using a realistic large dataset based on cloud environments for identifying network anomalies by exemplifying the detection of as many of the most common and modern attacks as possible that can occur via network communication. Our approach involved nine classification algorithms: KNN (using Manhattan and Euclidean distance), DTs (using Gini and Entropy), RF (using Gini and Entropy), Perceptron, Naïve Bayes, and Quadratic Discriminant Analysis classifiers. In our approach, we used the flow level to extract the features, which included a number of statistical variables that were affiliated to it. The classification algorithms were applied to 76 features and then by using PCA (n=2) to show how the least number of features affected by attacks and, consequently, to determine the effectiveness of the flow level in anomaly detection. From the results of this approach, we found that the performance of each classifier is different using PCA, compared to a similar accuracy when we used 76 features. The RF with entropy classifier achieved the highest accuracy rate with minimal training time when we combined all network traffic to a single file: 97.45 without PCA and 97%.4 with PCA. While we achieved 100% accuracy on all attacks except pivoting attacks from the results, we also concluded that some types of attacks, such as pivoting attacks, affected the accuracy. The overall conclusion is that using CSE-CIC-IDS-2018 data with ML techniques is a very powerful approach to stealth attack detection with a distinctive impact on cybersecurity in cloud environments.

APPENDIXES

Table 9: Results with PCA

	REnt			**RF-Gini**			**DT-Ent**			**DT-Gini**			**QDA**		
#	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
0	99%	99%	99%	100%	100%	100%	98%	98%	98%	99%	99%	99%	50%	68%	58%
1	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	92%	93%	93%
2	99%	100%	100%	100%	100%	100%	99%	99%	99%	99%	99%	99%	97%	95%	96%
3	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	93%	93%	93%
4	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	79%	8800%
5	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99%	99%	99%
6	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
7	82%	8300%	83%	82%	8300%	83%	78%	81%	80%	78%	81%	80%	89%	86%	89%
8	70%	72%	73%	70%	72%	73%	68%	70%	69%	68%	70%	69%	72%	72%	72%
9	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	83%	83%	83%
	GNB			**Perceptron**			**KNN-E**			**KNN-M**			*	*	*
#	P	R	F	P	R	F	P	R	F	P	R	F	*	*	*
0	26%	30%	28%	78%	81%	80%	100%	100%	100%	100%	100%	100%	*	*	*
1	92%	94%	93%	70%	95%	80%	100%	100%	100%	100%	100%	100%	*	*	*
2	97%	69%	80%	98%	95%	87%	100%	100%	100%	100%	100%	100%	*	*	*
3	91%	91%	91%	86%	86%	86%	100%	100%	100%	100%	100%	100%	*	*	*
4	84%	78%	81%	99%	73%	84%	100%	100%	100%	100%	100%	100%	*	*	*
5	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
6	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
7	89%	86%	89%	89%	86%	89%	88%	88%	88%	87%	87%	87%	*	*	*
8	72%	72%	72%	52%	52%	52%	75%	75%	75%	75%	75%	75%	*	*	*
9	60%	60%	60%	60%	60%	60%	100%	100%	100%	100%	100%	100%	*	*	*

Table 10: Results without PCA.

	RF-Ent			**RF-Gini**			**DT-Ent**			**DT-Gini**			**QDA**		
#	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
0	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
1	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
2	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
3	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
4	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
5	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
6	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	95%	91%	93%
7	83%	84%	83%	83%	84%	83%	78%	71%	80%	78%	71%	80%	30%	30%	30%
8	74%	76%	75%	74%	76%	75%	69%	71%	70%	69%	71%	70%	38%	38%	38%
9	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	GNB			**Perceptron**			**KNN-E**			**KNN-M**			*	*	*
#	P	R	F	P	R	F	P	R	F	P	R	F	*	*	*
0	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
1	85%	90%	87%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
2	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	
3	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
4	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
5	92%	80%	86%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
6	64%	52%	57%	100%	100%	100%	100%	100%	100%	100%	100%	100%	*	*	*
7	14%	14%	14%	79%	79%	79%	87%	87%	87%	87%	87%	87%	*	*	*
8	40%	40%	40%	66%	66%	66%	75%	75%	75%	75%	75%	75%	*	*	*
9	94%	94%	94%	99%	99%	99%	100%	100%	100%	100%	100%	100%	*	*	*

REFERENCES

1. O. Barayas., "**How Internet of Things (IoT) Is Changing the Cyber Security Landscape**," 2014. [Online]. Available: https://securityintelligence.com/how-the-internet-of-things-iot-ischanging-the-cybersecurity-landscape/. [Accessed: 14-Dec-2019].
2. C. Ventures, "**Cybercrime Infographics: Illustrations Of The Past, Present, And Future Threats We Face**," 2016. [Online]. Available: https://cybersecurityventures.com/cybercrime-infographic/. [Accessed: 14-Dec-2019].
3. R. K. Alqurashi, M. A. AlZain, B. Soh, M. Masud, J. Al-Amri. **Cyber Attacks and Impacts: A Case Study in Saudi Arabia**, International Journal of Advanced Trends in Computer Science and Engineering, vol. 9, no. 1, pp. 217–224, 2020 https://doi.org/10.30534/ijatcse/2020/33912020
4. M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "**Network Anomaly Detection: Methods, Systems and Tools**," IEEE Commun. Surv. TUTORIALS, vol. 16, no. 1, pp. 303–336, 2014.
5. P. Garc´ıa-Teodoro, J. D´ıaz-Verdejo, G. Macia -Fern´ a Ndez, and E.´ Va Zquez, "**Anomaly-based network intrusion detection: Techniques,´ systems and challenges**," Comput. Secur., vol. 28, pp. 18–28, 2009. https://doi.org/10.1016/j.cose.2008.08.003
6. D.-Y. Yeung, Y. Ding, and D.-Y. Yeung, "**Host-based intrusion detection using dynamic and static behavioral models**," Pattern Recognit., vol. 36, pp. 229–243, 2003.
7. D. Heckerman, "**A Tutorial on Learning with Bayesian Networks BT Learning in Graphical Models**," M. I. Jordan, Ed. Dordrecht: Springer Netherlands, 1998, pp. 301–354.
8. K. Sequeira and M. Zaki, "**ADMIT: Anomaly-based data mining for intrusions,**" in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 386–395. https://doi.org/10.1145/775047.775103
9. J. Patterson and A. Gibson, **Deep learning: a practitioner's approach**, 1st ed. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2017.
10. R. D. E. Ecosistemas, L. R. Bo, L. N. Mouso, D. A. Faggi, and L. A. Gainza, "**U Niversidad De B Uenos a Ires**," pp. 1–5, 2007.
11. R. Messier, **Network forensics**, 1st ed. Hoboken:Wiley, 2017.
12. A. Orebaugh, G. Ramirez, and J. Burke, **Wireshark Ethereal network protocol analyzer toolkit**, Ist. Rockland, MA; [Place of publication not identified] Syngress; Distributed by O'Reilly Media in the United States and Canada, 2007.
13. C. Perkins, "**IP Mobility Support for IPv4, Revised**," 2010. https://doi.org/10.17487/rfc5944
14. B. Claise, B. Trammell, and P. Aitken, "**Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information**," 2013.
15. M. Lucas, **Network flow analysis**, 1st ed. San Francisco: No Starch Press, 2010.
16. C. Bullard, "**Auditing Network Activity: Argus Layer 2+ Auditing Tool**," 2018. [Online]. Available: https://n0where.net/auditing-network-activity. [Accessed: 14-Dec-2019].
17. L. Metcalf and W. Casey, **Cybersecurity and applied mathematics**, 1st ed. Cambridge, MA: Syngress is an imprint of Elsevier, 2016.
18. E. W. Bethel, S. Campbell, E. Dart, K. Stockinger, K. Wu, and I. S. on V. A. S. and T. 2006, "**Accelerating Network Traffic Analytics Using Query-Driven Visualization**," in IEEE Symposium On Visual Analytics Science And Technology Baltimore-USA, 2006, pp. 115–122.
19. M. J. A. Qayyum, M. H. Islam, "**GENERIC MODEL FOR INTRUSION DETECTION**," in International Conference on Emerging Technologies (ICET), 2015, pp. 270–276.
20. S. Axelsson, "**Intrusion Detection Systems: A Survey and Taxonomy**," 2002.
21. D. E. Denning, "**An Intrusion-Detection Model**," 1987. https://doi.org/10.1109/SP.1986.10010
22. T. Dietterich, C. Bishop, D. Heckerman, M. Jordan, and M. Kearns, **Introduction to Machine Learning Second Edition Adaptive Computation and Machine Learning**, 2nd ed. MA: MIT Press, 2010.
23. K. Patel, "**Lowering the Barrier to Applying Machine Learning**," in CHI '10 Extended Abstracts on Human Factors in Computing Systems, 2010, pp. 2907–2910.
24. P. Barapatre, N. Z. Tarapore, S. G. Pukale, and M. L. Dhore, "**Training MLP neural network to reduce false alerts in IDS**," in 2008 International Conference on Computing, Communication and Networking, 2008, pp. 1–7.
25. C. Zhang, J. Jiang, and M. Kamel, "**Comparison of BPL and RBF Network in Intrusion Detection System BT - Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing**," 2003, pp. 466–470.
26. W. A. H. M. Ghanem and A. Jantan, "**A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm**," Neural Comput. Appl., vol. 4, no. 107, pp. 1–34, 2019.
27. C. Vaid and H. K. Verma, "**Anomaly-based IDS implementation in cloud environment using BOAT algorithm**," in Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, 2014, pp. 1–6.

28. A. Ammar, “**A Decision Tree Classifier for Intrusion Detection Priority Tagging**,” J. Comput. Commun., vol. 3, no. KSA, pp. 52–58, 2015. https://doi.org/10.4236/jcc.2015.34006
29. M. Nawir, A. Amir, O. B. Lynn, N. Yaakob, and R. Badlishah Ahmad, “**Performances of Machine Learning Algorithms for Binary Classification of Network Anomaly Detection System**,” J. Phys. Conf. Ser., vol. 1018, p. 12015, 2018.
30. P. Foremski, C. Callegari, and M. Pagano, “**Waterfall: Rapid Identification of IP Flows Using Cascade Classification BT - Computer Networks**,” in 21st International Conference on Computer Networks, 2014, pp. 14–23.
31. Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, “**Detection of denial-of-service attacks based on computer vision techniques**,” IEEE transactions on computers, vol. 64, no. 9, pp. 2519–2533, 2015.
32. E. A. Dmitriev and V. V Myasnikov, “**Comparative study of description algorithms for complex-valued gradient fields of digital images using linear dimensionality reduction methods**,” Comput. Opt., vol. 42, pp. 822–828, Sep. 2018.
33. J. Oliver, “**Jason brownlee - Deep learning with python**,” J. Chem. Inf. Model., vol. 53, no. 9, pp. 1689–1699, 2013. https://doi.org/10.1021/ci400128m
34. K. P. Murphy, **Machine Learning A Probabilistic Perspective**, 1sth ed. MIT press, 2012.
35. M. Chauhan, A. Pratap, Sonika, and A. Dixit, “**Designing a technique for detecting intrusion based on modified Adaptive Resonance Theory Network**,” in 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 2015, pp. 448–451.
36. M. Sabhnani and G. Serpen, “**Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context**,” in Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications, 2003, pp. 209–215.
37. P. Lichodzijewski, A. N. Zincir-Heywood, and M. I. Heywood, “**Hostbased intrusion detection using self-organizing maps**,” in Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN’02 (Cat. No.02CH37290), 2002, vol. 2, pp. 1714–1719 vol.2.
38. H. Gunes Kayacik, A. Nur Zincir-Heywood, and M. I. Heywood, “**A Hierarchical SOM-based Intrusion Detection System**,” Eng. Appl. Artif. Intell., vol. 20, no. 4, pp. 439–451, Jun. 2007.
39. M. Lichman, “**UCI Machine Learning Repository**,” 2013.
40. R. Kim, S. Pyke, “**DETECTING HACKERS (ANALYZING NETWORK TRAFFIC) BY POISSON MODEL MEASURE**,” 2004. .
41. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “**A Detailed Analysis of the KDD CUP 99 Data Set,**” in CISDA’09: Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications, 2009, pp. 1–6.
42. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “**Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization**,” in 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), 2018, no. Cic, pp. 108–116. https://doi.org/10.5220/0006639801080116
43. M. Lichman, “**1999 DARPA Intrusion Detection Evaluation Dataset — MIT Lincoln Laboratory**,” 2000. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusio n-detectionevaluation-dataset. [Accessed: 14-Dec-2019].
44. G. Shmoo, “**Public PCAP files for download**,” 2011. [Online]. Available: https://www.netresec.com/?page=PcapFiles. [Accessed: 14-Dec-2019].
45. CAIDA, “**CAIDA Data - Overview of Datasets, Monitors, and Reports**,” 2011. [Online]. Available: http://www.caida.org/data/overview/. [Accessed: 14-Dec-2019].
46. UNIBS, “**UNIBS: Data sharing**,” 2009. [Online]. Available: http://netweb.ing.unibs.it/ ntw/tools/traces/. [Accessed: 17-Dec-2019].
47. S. Bafandeh, I. And, and M. Bolandraftar, “**Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background**,” 2013.
48. H. Gao, F. Cao, and P. Zhang, “**Annulus: A novel image-based CAPTCHA scheme**,” IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON, pp. 464–467, 2017.
49. Z. Zhang, “**Introduction to machine learning: K-nearest neighbors**,” in Annals of Translational Medicine, vol. 4, no. 11, 2016, pp. 1–7.
50. H. Sarvari and M. M. Keikha, “**Improving the accuracy of intrusion detection systems by using the combination of machine learning approaches**,” in Proceedings of the 2010 International Conference of Soft Computing and Pattern Recognition, SoCPaR 2010, 2010, pp. 334–337. https://doi.org/10.1109/SOCPAR.2010.5686163
51. A. M. Saxe, J. C. McClelland, A. Y. Ng, K. V Shenoy, S. University., and D. of E. Engineering., “**Deep linear neural networks: a theory of learning in the brain and mind**,” Stanford, 2015.
52. A. Pena Ya˜ nez, “**El anillo esof˜ agico inferior.**” Rev. Esp. Enferm. Apar.´ Dig., vol. 26, no. 4, pp. 505–516, 1967.
53. B. G. Atli, “**Anomaly-Based Intrusion Detection by Modeling Probability Distributions of Flow Characteristics**,” Aalto University, 2017.
54. M. H. Kondarasaiah and S. Ananda, **Kinetic and mechanistic study of Ru(III)-nicotinic acid complex**

formation by oxidation of bromamine-T in acid solution, vol. 27, no. 1. 2004.

55. D. D. Gutierrez, **Machine learning and data science: an introduction to statistical learning methods with R. Basking Ridge**, N.J: Technics Publications, 2015.
56. J. Lampinen, J. Laaksonen, and E. Oja, “**Neural Network Systems, Techniques and Applications in Pattern Recognition**,” 1997. [Online]. Available:http://zeus.hut.fi/publications/ps/b1 nnsystems.ps. [Accessed: 14-Dec-2019].
57. P. Kaviani and S. Dhotre, “**Short Survey on Naive Bayes Algorithm**,” Int. J. Adv. Res. Comput. Sci. Manag.,vol. 04, Nov. 2017.
58. B. Ghojogh and M. Crowley, “**Linear and Quadratic Discriminant Analysis: Tutorial**,” 01-Jun-2019. [Online]. Available: https://danielmiessler.com/study/tcpdump.
59. A. Kalliola et al., “**Learning Flow Characteristics Distributions with ELM for Distributed Denial of Service Detection and Mitigation BT Proceedings of ELM-2016**,” in In ELM-2016 Cham-Switzerland, 2018, pp. 129–143.
60. V. Kanimozhi and T. P. Jacob, “**Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing**,” ICT Express, vol. 5, no. 3, pp. 211–214, 2019.
61. V. Kanimozhi and T. Prem Jacob, “**Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing**,” in Proceedings of the 2019 IEEE International Conference on Communication and Signal Processing, ICCSP 2019, 2019, pp. 33–36. https://doi.org/10.1109/ICCSP.2019.8698029
62. Q. Zhou and D. Pezaros, “**Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection – An Analysis on CIC-AWS-2018 dataset**,” 2019. [Online]. Available: http://arxiv.org/abs/1905.03685.
63. Darus, Mohamad Yusof, Mohd Afham Omar, Mohd Farihan Mohamad, Zulhairi Seman, and Norkhusahini Awang. **Web Vulnerability Assessment Tool for Content Management System**. International Journal 9, no. 1.3, 2020