International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 6 Issue 2, February - 2022, Pages:103-119

Evaluating Performance of Supervised Learning Techniques for Developing Real-Time Intrusion Detection System

Shadi I. Abudalfa, Ekhlas S. Isleem, Marah J. Elshaikh Khalil, Rewaa A. Dalloul and Shaimaa M. Iqtefan

Computer Engineering Department
University College of Applied Sciences, Gaza, Palestine
sabudalfa@ucas.edu.ps

Abstract— *Nowadays, a lot of organizations are currently suffering from serious security threats. Those threats often keep changing and may evolve to new, dangerous, and unknown types. Thereby, such organizations suffer from a business problem that consumes a huge cost for detecting a tremendous number of threats. For tackling this problem, intrusion detection systems have been presented to automatically detect security threats. Some of these systems use machine learning techniques for provides high accuracy with detecting new attacks. In this paper, we focus on developing intrusion detection system that uses supervised learning technique. The performance of several machine learning techniques were evaluated for highlighting the best technique that can be used for implementing intrusion detection systems. As a result of this work, a comparison framework has been provided by using public dataset. The techniques were evaluated by measuring classification Accuracy, Recall, Precision and F1-Score with different categories of attacks. The experimental results show that the highest accuracy is about 96.974% with using Decision Tree (DT) technique.*

Keywords— Machine Learning; Intrusion Detection; Attack; Threat

1. Introduction

With the development of the internet and its wide applications in all domains of everybody's life, cybersecurity is highly needed to keep the network safe against various types of cyber-attacks that appear daily. There are many approaches proposed for securing computer networks, one of which is based on detecting network attacks then generating an alert when catching malicious behavior which is called intrusion detection system (IDS). The intrusion detection system has become an essential part of network security of monitoring incoming network traffic and attempting to recognize and notify it as either normal or abnormal.

There are two main types of IDS: network intrusion detection system (NIDS), which identifies intrusions by examining network traffic and monitors multiple hosts, and host-based intrusion detection system (HIDS), which identifies intrusions by analyzing system calls, application logs, file-system modifications and other host activities. All intrusion detection systems use one of two detection techniques: signature-based IDS, which detects malicious traffic based on comparing with known-attack signatures, and anomaly-based (behavioral) IDS which aims to detect abnormal behaviors.

The traditional IDS still suffers from a high false alarm rate, generating many alerts for low non-threatening situations, which raises the burden for security analysts and can cause a seriously harmful attack to be ignored. Another problem with existing IDSs is a low ability to detect unknown attacks since network environments change quickly and new attacks emerge constantly. Thus, there is a need to develop IDS that provides higher detection rates and reduces false alarm rates along with detecting unknown attacks.

Recently, Machine Learning (ML) techniques have gained wide interest in tasks of intrusion detection to improve traditional IDS. The resulted system is based on building a model learned by using labeled dataset. In this work, we investigate this issue by evaluating performance of different machine learning techniques by applying them to a public dataset used for implementing dynamic IDS.

The rest of the paper is organized as follows: Section 2 introduces theoretical background and describes evaluation metrics. Section 3 presents a survey for IDSs and related research works. Section 4 covers all details of the methodology used for evaluating the selected machine learning techniques and developing a real-time IDS system. Section 5 illustrates analysis of experimental results. Finally, Section 6 concludes the paper and presents suggestions for future work.

2. Background

This section presents a brief background about intrusion detection system (Definition, Types, and Techniques), explain famous types of attacks that we deal with in this research. We also present a background for machine learning techniques used with our work. Additionally, the evaluation measures are described in the end of this section.

2.1 Types of Intrusion Detection System (IDS)

Intrusion Detection System (IDS) is the process of monitoring for and identifying attempted unauthorized system access or manipulation. There are four main types of IDS [1]: Network Intrusion Detection System (NIDS), Host-based Intrusion Detection System (HIDS), Perimeter Intrusion Detection System (PIDS) and VM-based Intrusion Detection System (VMIDS). Our work is based on NIDS which monitors network traffic. Network intrusion detection systems gain access to network traffic by connecting to a network hub, either a network switch configured for port mirroring, or a network tap.

HIDS includes an agent that detects intrusion through analyzing the host activities. PIDS detects and pinpoints the location of intrusion attempts on perimeter fences of critical infrastructures. It uses either electronics or more advanced fiber optic cable technology fitted to the perimeter fence. The PIDS detects disturbances on the fence, and if an intrusion is detected, an alarm will be triggered. VMIDS detects intrusions using virtual machine monitoring. It is the most recent type and it is still under development.

2.2 Types of Network Attacks

In this subsection, we briefly describe the attacks checked with this research by using public dataset [2].

- Brute force attack is a common network attack type where attackers try to guess online passwords using every key combination or try to check if certain hidden web page.
- Denial of Service (DoS) attack is a very common type of network attack where attackers target a service or network by sending an overwhelming number of bogus requests to temporarily deny the legitimate users.
- Distributed Denial of Service (DDoS) is more sophisticated denial of service attack where attackers use multiple botnets consisting of tens of thousands of compromised systems to flood the victimized systems by network traffic.
- Bot attack is carried out by attackers that use group of compromised network systems and devices working as a single virtual network to carry out various nefarious Internet attacks.
- Web attack consists of three common attacks: cross-site scripting (XSS) (BruteForce-XSS), SQL-Injection, brute force administrative and user passwords (BruteForce-Web), etc. on modern web applications. For example, Damn Vulnerable Web App (DVWA) was used as a victim service and homegrown automated code using Selenium framework was used as an attacking tool to generate Web attack traffic.
- Infiltration attacks are usually carried out from inside the networks once some systems are compromised by exploiting critical vulnerabilities in software applications. After successful attacks, these systems provide persistent backdoors thereby enabling attackers to launch further internal attacks.

2.3 Machine learning Techniques

In this work, we explored the data and run several classification models. Then, we go back to the data and repeat the process until selecting a classification algorithm [3] that provide the best performance. We evaluated performance of four supervised learning techniques: Decision Trees, Random forest, Naive Bayes, Multi-Layer Perceptron and Logistic Regression. Next, we describe them briefly.

- Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural network model that consist of at least three layers of nodes: an input layer, a hidden layer and an output layer. The nodes are neurons that use a nonlinear activation function.
- Decision Trees (DT) use training data for creating data structure of tree. Then, the tree used for making predictions on the test data. This technique is based on providing accurate result with the least number of the decisions.
- Naive Bayes Classifier (NB) is based on Bayes’ theorem, with an assumption that each feature is independent from one another. When the data is continuous, the assumption is that the continuous values for each class are distributed according to a Gaussian distribution. Naive Bayes is generally used to represent the class conditional probability for continuous attributes.
- Random Forest (RF) is a popular ensemble method based on a standard machine learning technique called "decision tree". In a decision tree, a pair of variable-value is chosen that will split in such a way to generate “best” two child subsets. Then, the algorithm is applied on each branch of the tree. The input enters the root of the tree and traverses down the tree and gets bucketed into smaller sets. Random forest chooses a subsample of the feature space in each split and makes the trees de-correlated.
- Logistic regression (LR) is a probabilistic statistical classification model. It is a simple method estimates a linear best fitting model. The based predictor variable is formed based on combination of values taken by the predictors.

2.4 Performance Measures

We evaluated the performance selected classifiers by using five evaluation metrics: confusion matrix analysis, Classification Accuracy, Precision, Recall and F1-Score [4].

The confusion matrix shows a table that visually describes a summary of prediction results. The diagonal values report clorretly classified outcomes.

Classification Accuracy estimates the ratio of the correctly recognized connection records to the entire test data. If the accuracy is higher, the machine learning model is better. It is calculated as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \qquad (1)$$

Where, TP denotes to true positive samples. TN represents true negative samples. FP and FN are abbreviations to false positive and false negative samples respectively.

Precision estimates the ratio of the correctly identified attack connection records to the number of all identified attack connection records. If the Precision is higher, the machine learning model is better.

$$\text{precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad (2)$$

Recall which is also called True Positive Rate (TPR) estimates the ratio of the correctly classified Attack connection records to the total number of Attack connection records. If the TPR is higher, the machine learning model is better

$$\text{TPR} = \frac{\text{TP}}{\text{TP+FN}} \qquad (3)$$

F1-Score which is also called F1-Measure represents the harmonic mean of Precision and Recall. If the F1-Score is higher, the machine learning model is better

$$\text{F1} - \text{Score} = 2 \times \left(\frac{\text{Precision}\times\text{Recall}}{\text{Precision +Recall}}\right) \qquad (4)$$

3. LITERATURE REVIEW

We surveyed two main aspects in order to get our research aim. The first aspect leads us to survey several approaches to study and evaluate machine learning techniques used for intrusion detection. The second aspect is to discover available systems developed for detecting intrusions.

3.1 Related Works

In this subsection, we present some related works that have been achieved to employ machine learning for developing IDS. Next, we describe some of them. While, Table 1 shows the rest of selected related works. The table lists the title of the paper, the used dataset, year of publication and the used machine learning techniques.

Gobinath Loganathan et. al. [5] used encoder-decoder and recurrent neural network models for predicting a network packet sequence based on previous packets. This model is trained on an attack-free dataset to learn the normal sequence of packets in TCP connections. Then, it is used to detect anomalous packets in TCP traffic. They apply this model to DARPA 1999 dataset. Thereby they used an old data that does not have enough attacks. Their work reports accuracy with 97%.

Ahmed Ahmim et al [6] combined different classifier approaches based on decision tree and rules-based concepts to classify the network traffic as Attack/Benign. Their presented IDS used the CICIDS2017 dataset that includes unbalanced data since benign records are larger than attack records. Their experimental results report accuracy with 96.665%.

Nathan Shone et al [7] used deep learning model to enable NIDS operation. They performed extensive evaluations by using KDD Cup '99 and NSL-KDD datasets. Thereby, they also used very old datasets for developing IDS. Additionally, Nguyen Thanh Van et al [8] used deep learning techniques to implement an anomaly based NIDS for detecting unknown attacks. They used kddcup99 dataset which includes old data for detecting attacks of four groups: DoS, Probe, U2R and R2L.

Peng Lin et al. [9] built dynamic anomaly NIDS by using deep learning methods. They used the CSE-CICIDS2018 dataset which includes seven different attack scenarios such as DDoS attack, Botnet attack, Infiltration attack, Brute Force attack, Dos attack, Web attack, and Heart leech. Their experiment work reported accuracy with 96.2%.

Some research works used unsupervised machine learning techniques [10][11] for developing IDS. For example, Meenal Jain et al [12] employed k-means clustering algorithm in a distributed framework for detecting attacks. They also evaluated performance of decision tree and random forest classification techniques. A real time anomaly detection dataset ISCX 2012 has been used to analyze the effectiveness of the presented techniques.

Based on our literature review, we can say that most of available works used old datasets with limited attack types. Fig. 1 summaries all attacks detected with the selected research works. To fill this gap we used new dataset for conducting our experiment work. We used some well-known techniques to mimic other related works. Fig. 2 shows a summary of machine learning techniques used with the selected works.

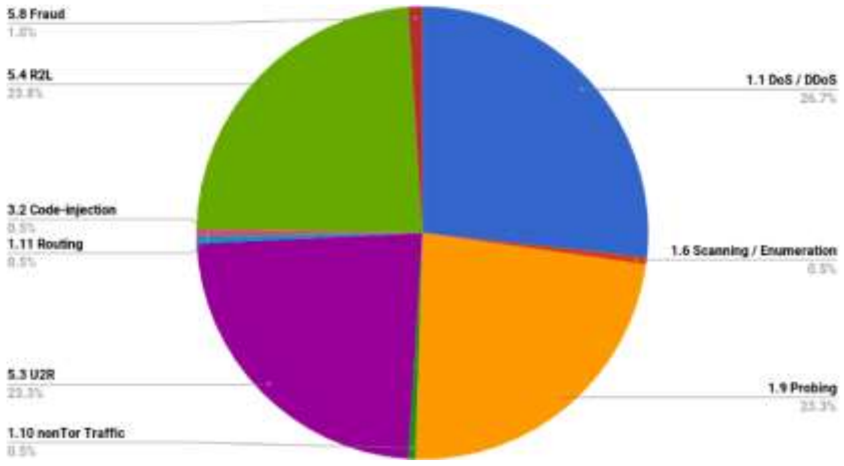


Fig. 1. *Summary of attacks detected by related works.*

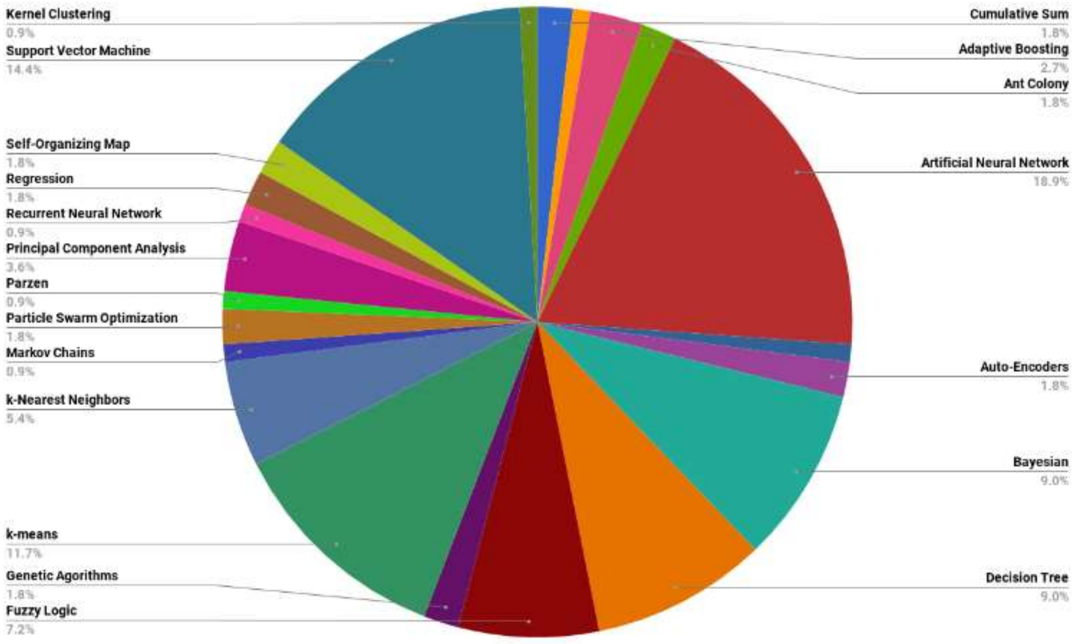


Fig. 2. *Summary of machine learning techniques used with related works.*

Table 1: Comparison of related works

Paper	Attacks	Dataset	ML Techniques

Support Vector Machine for Network Intrusion and Cyber-Attack Detection (2017)[13]	De authentication & injection attack using Airpwn for attacks LAN: Port scanning (probing) using Nmap	Collected datasets	SVM*
A Novel Distributed Semi-Supervised Approach for Detection of Network-Based Attacks (2019)[12]	Network attacks	NSL-KDD	CL*, DT*&RF*
A Deep Learning Approach for Intrusion Detection System in Industry Network (2017)[14]	Dos, network attack	Collected dataset	ANN* & DL*
Machine Learning Techniques used for the Detection and Analysis of Modern Types of DDoS Attacks (2017)[15]	The modern type of DDoS attacks such as SIDDoS, HTTP flood, Smurf, UDP flood.	Collected dataset	BN*, SVM* &DT*
A Convolutional Neural Network for Network Intrusion Detection System (2018)[16]	Network attacks	NSL-KDD	ANN* & DL*
A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks (2017)[17]	Host and Network Attack	Collected dataset	ANN* & DL*
Big Data Analytics for Network Intrusion Detection (2017)[18]	Network Attack	Collected dataset	ANN*,BN*,SVM* &CL*
Deep learning approach for Network Intrusion Detection in Software Defined Networking (2016)[19]	web attack	Collected dataset	DL*
Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection (2018)[20]	Zero-Day	CIC-AWS-2018 dataset	DT*
Network Intrusion Detection System Using Machine Learning (2018)[21]	web attack, brute force, DoS, DDoS, infiltration, Bot	CICIDS 2017 dataset	DT*
Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning (2019)[22]	DoS/DDoS attack	CSE-CIC-IDS2018	RF* & DL*
Real-time Intrusion Detection using Multidimensional Sequence-to-Sequence Machine Learning and Adaptive Stream Processing (2018)[23]	FTP Brute Force attack, Slow Header DoS attack, HTTP Unbearable Load King (HULK) DoS attack, SQL Injection attack, Web Brute Force attack, Cross-	CICIDS 2017 dataset	DL*

	site scripting attack, Ares Botnet attack, and Port Scan		

***ANN**: Artificial Neural Networks, ***BN**: Bayesian Networks, ***SVM**: Support Vector Machine, ***CL**: Clustering (e.g., k-Means and k-Nearest Neighbours), ***DT:** Decision Trees, ***RF:** Random Forests. ***NIDS:** Network Intrusion Detection Systems, ***DL**: Deep Learning.

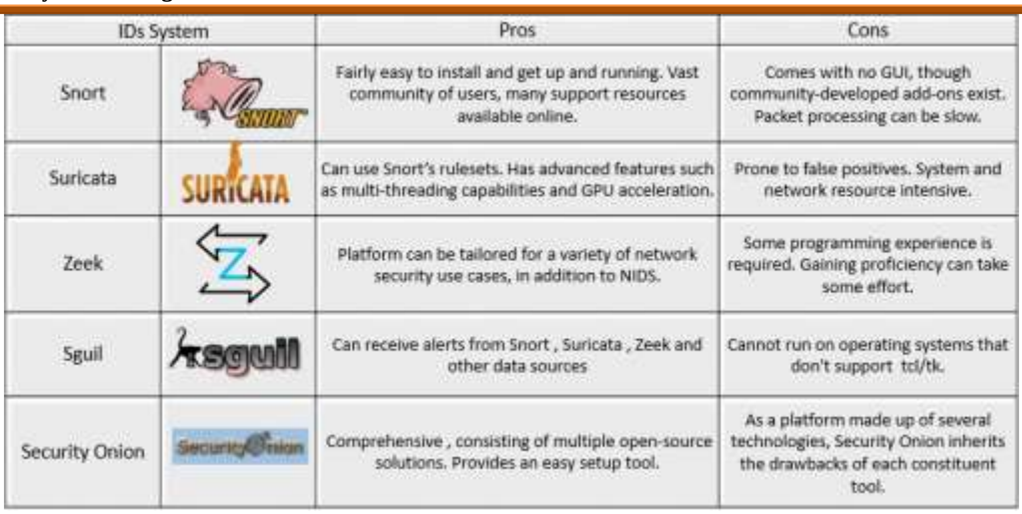
3.2 Intrusion Detection Systems

In this subsection, we describe some of the most popular IDSs. Table 2 illustrates a comparison between our presented system and other related systems. Next, we describe some of them briefly. A short comparison is illustrated as well in Fig. 3 to show the differences. Table 3 presents a comparison of all selected related systems by reporting type of IDS, the developer, free or not, the used language and the supported operating system.

- Snort is an open-source network intrusion prevention system (IPS) developed by Cisco. It performs real-time traffic analysis and packet logging. Additionally, it performs protocol analysis, content searching and matching. Snort is used as a straight packet sniffer, a packet logger and a full-blown network intrusion prevention system.
- Suricata is an open source IDS that exploits remotely created run sets to screen sniffed arranges activity and gives alarms when suspicious occasions occur. Suricata supports a multithreaded technique. The Suricata engine is capable to apply real-time detection, inline intrusion prevention (IPS), network security monitoring (NSM) and offline pcap processing.
- Wazuh is an open-source and enterprise-ready security monitoring solution for threat detection, integrity monitoring, incident response, and compliance. It is used to collect, aggregate, index and analyze security data, helping organizations detect intrusions, threats, and behavioral anomalies. Wazuh agent runs at a host-level (HIDS), combining anomaly and signature-based technologies to detect intrusions or software misuse.
- Zeek (Bro) is a NIDS that operates at the Application Layer. It supports a signature-based system and anomaly-based detection methods. It can spot bit-level patterns that indicate malicious activity across packets.

Table 2: Comparing the presented IDS with other related works.

Feature / System	Ours	Snort	Suricata	Bro (Zeek)
Easy management system	✓		✓	
Free & open Source	✓	✓	✓	✓
Detection Techniques	Anomaly	Signature	Signature	Signature & Anomaly
Prevention		✓	✓	
Log Supported	✓	✓	✓	✓
Difficult installation process				✓
Real Time	✓	✓	✓	✓
System Supported	Linux	Windows, FreeBSD, Linux, and Unix	FreeBSD, Linux, UNIX, Mac OS and windows	Linux, FreeBSD, and Mac OS
Have GUI	✓		✓	
Detection of multiple and different attack types	✓	✓	✓	



IDs System		Pros	Cons
Snort		Fairly easy to install and get up and running. Vast community of users, many support resources available online.	Comes with no GUI, though community-developed add-ons exist. Packet processing can be slow.
Suricata		Can use Snort's rulesets. Has advanced features such as multi-threading capabilities and GPU acceleration.	Prone to false positives. System and network resource intensive.
Zeek		Platform can be tailored for a variety of network security use cases, in addition to NIDS.	Some programming experience is required. Gaining proficiency can take some effort.
Sguil		Can receive alerts from Snort , Suricata , Zeek and other data sources	Cannot run on operating systems that don't support tcl/tk.
Security Onion		Comprehensive , consisting of multiple open-source solutions. Provides an easy setup tool.	As a platform made up of several technologies, Security Onion inherits the drawbacks of each constituent tool.

Fig. 3. *Comparison of selected IDSs.*

Table 3: Comparison of Different IDSs

IDS	Type	Developed by	free or not	Lang	Releases	Installed on
Snort [24]	NIDS / IPS*	Cisco	free open source	C	August 29, 2018	Windows, Linux, and Unix
SolarWinds Security [25]	HIDS*, but you can use NIDS* functions		not free but Free 30 Day Trial			Windows Server, it can collect data from Linux, Unix, and Mac OS
Suricata [26]	NIDS* that operates at the Application Layer	OISF	open-source software	C, Rust	April 30, 2019	FreeBSD, Linux, UNIX, Mac OS and Windows
Sagan [27]	HIDS*	Quadrant Information Security	open source	C	3 July 2019	Linux and Mac
OSSEC [28]	HIDS*	Daniel B. Cid	free and open-source		April 19, 2019	Linux, OpenBSD, FreeBSD, OS X, Solaris and Windows
ACARM-ng [29]	NIDS and HIDS / IPS*	WCSS	open source	C++ Python	29 May 2012	Linux
Fail2Ban [30]	IDS / IPS*	Cyril Jaquier et al[20]		Python	October 4, 2018	Unix-like
Samhain [31]	HIDS*	Samhain Services	open-source	C	January 7, 2019	Linux, all POSIX/UNIX Systems
Zeek [32]	NIDS*	Vern Paxson	free and open-source software	C++	September 23, 2019	Linux, FreeBSD, macOS
Firestorm [33]	NIDS*		Free		2013-02-21	
Chkrootkit [34]	HIDS*	Pangeia Informatica	free and open-source software	C	Feb 11, 2019	Linux, FreeBSD, OpenBSD, NetBSD, Solaris, HP-UX, Tru64, BSD/OS, Mac OS X

Tripwire [35]	HIDS*	Tripwire, Inc.	free and open-source software	C++, Perl	31 March 2018	Linux, all POSIX/UNIX Systems

***HIDS :** Host Intrusion Detection System ***NIDS:** Network Intrusion Detection System ***IDS:** Intrusion Detection System ***IPS:** Intrusion Prevention System

4. METHODOLOGY

This section covers the explanation details of methodology and dataset that used for achieving this work. The methodology is based on providing the best result during performance evaluation of the selected techniques. We also describe the experiment environment that used for reporting all results.

4.1 Dataset

In the subsection, we describe CSE-CIC-IDS2018 dataset along with included features. We also explain data processing methods used for preparing training and testing data. The used dataset is publically available for researchers. It is collected through a collaborative project achieved by the Communications Security Establishment (CSE) and The Canadian Institute for Cybersecurity (CIC). This dataset includes a detailed description of intrusions along with abstract distribution models for applications, protocols, or lower level network entities. The dataset includes seven attack scenarios: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. The used attacking infrastructure includes 50 machines and the victim organization has 5 departments includes 420 PCs and 30 servers [36].

The dataset includes the network traffic and log files of each machine from the victim side, along with 80 network traffic features extracted from captured traffic by using CICFlowMeter-V3 tool [37]. The included features with CIC-AWS dataset are described in Table 4.

Table 4: Features used with the dataset

Feature	Description	Type
Label	indicate whether the traffic is malicious or not, e.g., benign, SQLInjection, etc.	string
Dst Port	Destination port number	integer
Protocol	Protocol	integer
TimeStamp	Time Stamp of the flow	string
Flow Duration	Flow duration	integer
Tot Fwd/BwdPkts	Total packets in forward/backward directions	integer
TotLenFwd/BwdPkts	Total size of packets in forward/backward directions	integer
Fwd/BwdPkt - Len Max/Min/Mean/Std	Maxi/Mini/Average/Std. Dev. size of package in forward/backward directions	integer
Flow Byts/s & Flow Pkts/s	Flow byte rate, i.e., number of packets per seconds	float64
Flow IAT Mean/Std/ Max/Min	Average/Std. Deviation/Maxi/Mini time between two flows	float64
Fwd/Bwd IAT Tot/Mean/ Std/-Max/Min	Total/Average/Std. Deviation/Maxi/Mini time between two packets in forward/backward directions	float64
Fwd/Bwd PSH/URG Flags	Number of times the PSH/URG flag was set in packets in forward/backward direction	integer
Fwd/Bwd Header Len	Total bytes used for header in forward/backward direction	integer
Fwd/BwdPkts/s	Number of forward/backward packets per second	float64
Pkt Len Min/Max/Mean/Std	Maxi/Mini/Average/Std. Dev. length of a flow	integer
Pkt Len Var	Mini inter-arrival time of packet	float64
FIN/SYN/RST/PUSH/ACK/ URG/CWE/ECE Flag Cnt	Number of packets with FIN/SYN/RST/PUSH/ACK-/URG/CWE/ECE	integer
Down/Up Ratio	Download/upload ratio	integer
Pkt Size Avg	Average size of packets in forward/backward direction	float64
Fwd/BwdSeg Size/Byts/b/Blk Rate Avg	Average number of bulk rate/bytes bulk rate/packets bulk rate in forward/backward directions	float64
SubflowFwd/BwdPkts/Byts	The average number of bytes/packets in a sub flow in forward/backward direction	integer
InitFwd/Bwd Win Byts	Number of bytes sent in initial window in forward/backward directions	integer

Fwd Act Data Pkts	Number of packets with at least 1 byte of TCP data payload in forward	integer
FwdSeg Size Min	Minimum segment size observed in forward	integer
Active Mean/Std/Max/Min	Maxi/Mini/Average/Std. Dev. a flow was active before becoming idle	float64
Idle Mean/Std/Max/Min	Maxi/Mini/Average/Std. Dev. a flow was idle before becoming active	float64

The dataset contains15,450,706 rows divided into 10 files. While, each row has 80 features. The content of these files is described in Table 5. We have collected the dataset files and merged all the classes related to a specific attack with each other in one CSV file to apply the machine learning techniques. As Shown in Table 6, the dataset includes (benign) and six common attack types: Brute-force, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside.

Table 5: The description of dataset files

File Name	Contents
File 1 "Wednesday-14-02-2018"	It contains benign traffic (667,626 rows) and Two types of brute-force attacks: SSH-Bruteforce (187,589 rows). FTP-BruteForce (193,360 rows).
File 2 "Thursday-15-02-2018"	It contains benign traffic (996,077 rows) and two types of DoS attacks: DoS attacks-Slowloris (10,990 rows). DoS attacks-GoldenEye (41,508 rows).
File 3 "Friday-16-02-2018"	It contains benign traffic (442,020 rows) and two types of DoS attacks: DoSattacks-Hulk (466,664 rows). DoS attacks-SlowHTTPTest (139,890 rows).
File 4 "Thursday-20-02-2018"	It contains benign traffic (7,372,557 rows) and one type of DDoS attack: DDOS attack-LOIC-HTTP (576,191 rows).
File 5 "Wednesday-21-02-2018"	It contains benign traffic (360,833 rows) and two types of DDoS attacks: DDOS attack-HOIC (686,012 rows). DDOS attack-LOIC-UDP (1730 rows).
File 6 "Thursday-22-02-2018"	It contains benign traffic (1,048,213 rows) and three types of web attacks: SQL Injection (34 rows). Brute Force -Web (249 rows). Brute Force –XSS (79 rows).
File 7 "Friday-23-02-2018"	It contains benign traffic (1,048,009 rows) and three types of web attacks: SQL Injection (53 rows). Brute Force -Web (249 rows). Brute Force –XSS (151 rows).
File 8 "Wednesday-28-02-2018"	It contains benign traffic (544,200 rows) and one type of infiltration attack: Infiltration (68,871 rows).
File 9 "Thursday-01-03-2018"	It contains benign traffic (238,037 rows) and one type of infiltration attack: Infiltration (93,063 rows).
File 10 "Friday-02-03-2018"	It contains benign traffic (762,384 rows) and one type of Botnet attack: Bot (286,191 rows).

As shown in Table 6, the ratio between benign and malicious network traffic is heavily skewed in favor of benign traffic which account about 83% of all data. The dataset contains a high amount of features with a strong correlation, suggesting that many of those features are redundant and can be removed. We consider this issue when performing data processing. Fig. 4 shows some examples of features with strong correlation.

Table 6: Traffic data of each attack type included in the dataset

Category	**Attack Type**	**Number of Samples**
Brute-force	SSH-Bruteforce	187,589
	FTP-BruteForce	193,354
Web attack	Brute Force –XSS	230
	Brute Force –Web	611
	SQL Injection	87
DoS attack	DoS attacks-Hulk	466,664
	SlowHTTPTest	139,890
	DoS attacks-Slowloris	10,990
	DoS attacks-GoldenEye	41,508
DDoS attack	DDOS attack-HOIC	686,012
	DDOS attack-LOIC-UDP	1730
	DDOS attack-LOIC-HTTP	576,191
Botnet	Bot	286,191
Infilteration	Infilteration	161,934
Benign		12,697,719
Total		15,450,706

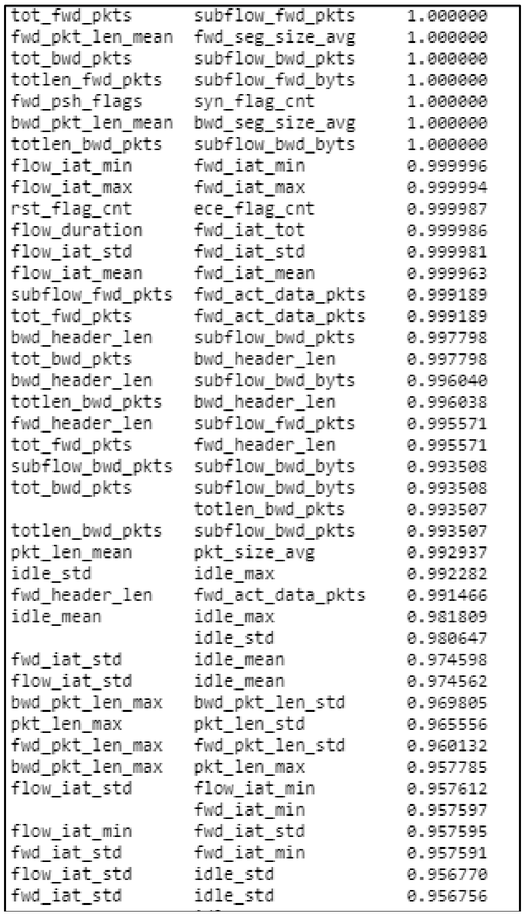

```
tot_fwd_pkts       subflow_fwd_pkts     1.000000
fwd_pkt_len_mean   fwd_seg_size_avg     1.000000
tot_bwd_pkts       subflow_bwd_pkts     1.000000
totlen_fwd_pkts    subflow_fwd_byts     1.000000
fwd_psh_flags      syn_flag_cnt         1.000000
bwd_pkt_len_mean   bwd_seg_size_avg     1.000000
totlen_bwd_pkts    subflow_bwd_byts     1.000000
flow_iat_min       fwd_iat_min          0.999996
flow_iat_max       fwd_iat_max          0.999994
rst_flag_cnt       ece_flag_cnt         0.999987
flow_duration      fwd_iat_tot          0.999986
flow_iat_std       fwd_iat_std          0.999981
flow_iat_mean      fwd_iat_mean         0.999963
subflow_fwd_pkts   fwd_act_data_pkts    0.999189
tot_fwd_pkts       fwd_act_data_pkts    0.999189
bwd_header_len     subflow_bwd_pkts     0.997798
tot_bwd_pkts       bwd_header_len       0.997798
bwd_header_len     subflow_bwd_byts     0.996040
totlen_bwd_pkts    bwd_header_len       0.996038
fwd_header_len     subflow_fwd_pkts     0.995571
tot_fwd_pkts       fwd_header_len       0.995571
subflow_bwd_pkts   subflow_bwd_byts     0.993508
tot_bwd_pkts       subflow_bwd_byts     0.993508
                   totlen_bwd_pkts      0.993507
totlen_bwd_pkts    subflow_bwd_pkts     0.993507
pkt_len_mean       pkt_size_avg         0.992937
idle_std           idle_max             0.992282
fwd_header_len     fwd_act_data_pkts    0.991466
idle_mean          idle_max             0.981809
                   idle_std             0.980647
fwd_iat_std        idle_mean            0.974598
flow_iat_std       idle_mean            0.974562
bwd_pkt_len_max    bwd_pkt_len_std      0.969805
pkt_len_max        pkt_len_std          0.965556
fwd_pkt_len_max    fwd_pkt_len_std      0.960132
bwd_pkt_len_max    pkt_len_max          0.957785
flow_iat_std       flow_iat_min         0.957612
                   fwd_iat_min          0.957597
flow_iat_min       fwd_iat_std          0.957595
fwd_iat_std        fwd_iat_min          0.957591
flow_iat_std       idle_std             0.956770
fwd_iat_std        idle_std             0.956756
```

Fig. 4. *Feature correlation.*

4.2 Data Preprocessing

The Following steps are adopted for applying data preprocessing to the used dataset:

1. Delete noisy features and missing value.
2. Format data into standard data type.
3. Reduce the size of the dataset by selecting part of it.
4. Drop samples with "Infinity" and "NaN" value.
5. Parse and remove columns that were repeated in the dataset.

After applying data preprocessing, about 36,366 samples were dropped as a result of the data cleanup process. Table 7 shows the number of selected dropped samples.

Table 7: Number of samples selected and dropped from the used dataset

Dataset	Traffic Type	Selected Samples #	Dropped Samples #
02-14-2018.csv	Benign	663,808	3,818
	FTP-BruteForce	193,354	6
	SSH-Bruteforce	187,589	0
02-15-2018.csv	Benign	988,050	8,027
	DoS-GoldenEye	41,508	0
	DoS-Slowloris	10,99	0
02-16-2018.csv	Benign	446,772	0
	DoS-SlowHTTPTest	139,890	0
	DoS-Hulk	461,912	0
02-22-2018.csv	Benign	1,042,603	5,610
	BruteForce-Web	249	0
	BruteForce-XSS	79	0
02-23-2018.csv	Benign	1,042,301	5,708
	BruteForce-Web	362	0
	BruteForce-XSS	151	0
	SQL-Injection	53	0
03-01-2018.csv	Benign	235,778	2,259
	Infiltration	92,403	660
03-02-2018.csv	Benign	758,334	4,050
	BotAttack	286,191	0

4.3 Machine Learning

We evaluated the selected machine learning technique by using 80% of data for training and 20% for testing. We also applied feature selection as follows. Thereby, the remaining number of features is 33 features.

1. All features with zero amount of variation are removed as those features will not have any influence on the prediction.
2. We removed undesired features such as Timestamp.
3. Features with high correlation are removed. We removed them since to decrease the effect of noise and redundancy.

The main task is getting a set of statistics information of traffic flow, identify whether this flow is benign traffic or intrusion based on using the training data. To achieve this task, we applied cross-validation algorithm with K=5 on the dataset. Then, we applied different supervised classification techniques to find the best model that suits our system. We evaluated five machine learning algorithms and compared their results by reporting Precision, Recall, F1-Score, and Accuracy. We used Random forest (RF), Naive Bayes (NB), Decision tree (DT), Multi-layer Perceptron classifier (MLP) and Logistic Regression (LR).

4.4 Real-Time Network Intrusion Detection System (NIDS)

To simulate the presented NIDS, an advanced simulation environment has been created for our research. This infrastructure is like a real network environment that allows us to test our model in semi real environment with traffics. In order to efficiently simulate real network infrastructure, we used Graphical Network Simulator (GNS3) software. GNS3 is a free graphical network emulator used for testing networks that built from virtual equipment. GNS3 can create complex network topologies based on virtual hardware. Our our network topology used with this research is shown in Fig. 5.

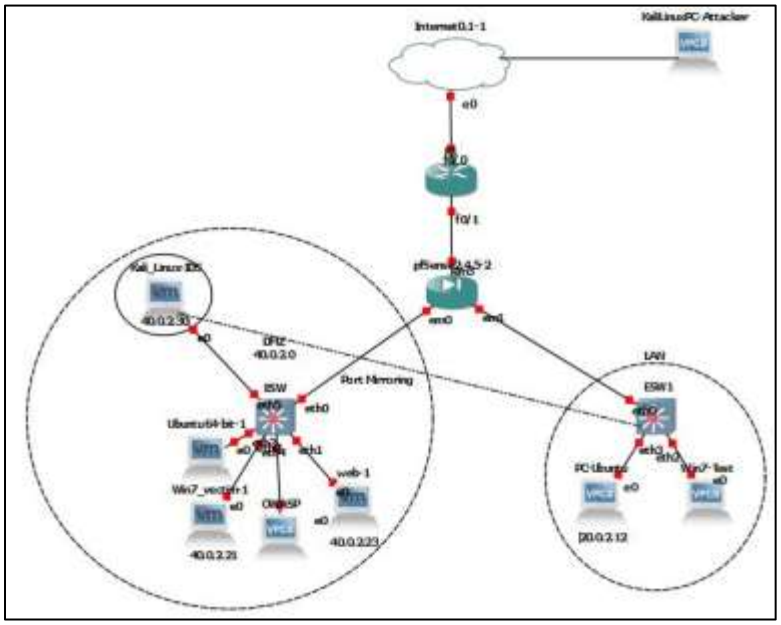


Fig. 5. *Network topology used for simulation.*

Pfsense firewall is used for filtering the incoming traffic and allows the outside network to get only the DMZ zone and the servers inside it and preventing them from getting to the LAN zone. The pfsense has three network adapters: WAN, LAN and DMZ networks. We enabled DHCP on all network adapters. The Lan network contains the private computers and devices that can connect to internet but cannot get accessed by outside. The DMZ network contains servers that connected with outside servers such as WEB and FTP.

Router R1 is used it to give the same functionality of the hardware router. 2 Router switch at layer3 (EtherSwitch Router) are used for port mirroring. They also known as switched port analyzer (SPAN) which is a method of copying and sending network packets transmitted as input from ports connected with switch to another port of the monitoring device. Thus, we implemented this monitoring technique on LAN and DMS to monitor the server that has our NIDS. We used Kali Linux and its offensive tools as attacker device. WEB server in the DMZ zone can be requested from outside networks. Windows 7 as one of the devices in the LAN network is used for generating some attacks.

Fig. 6 shows the design of the presented system. Monitor system (Tcpdump) is used to capture all traffic in the network and store it as PCAP files. We use CICFlowMeter to convert PCAP file to CSV file and extract the features. The CSV files represent the dataset that needs preprocessing before applying the classifier. IDS system will classify traffic in the CSV files by using the classifier and detect attack if exists. In detection module, if the traffic was seen as malicious, then the IDS system will make an alert and generate a log and store it in SQLite.

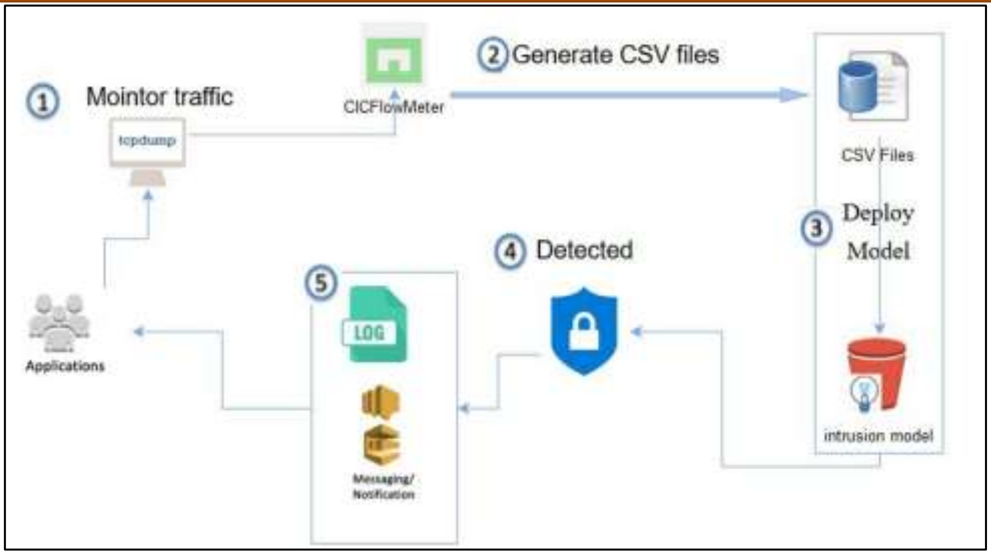
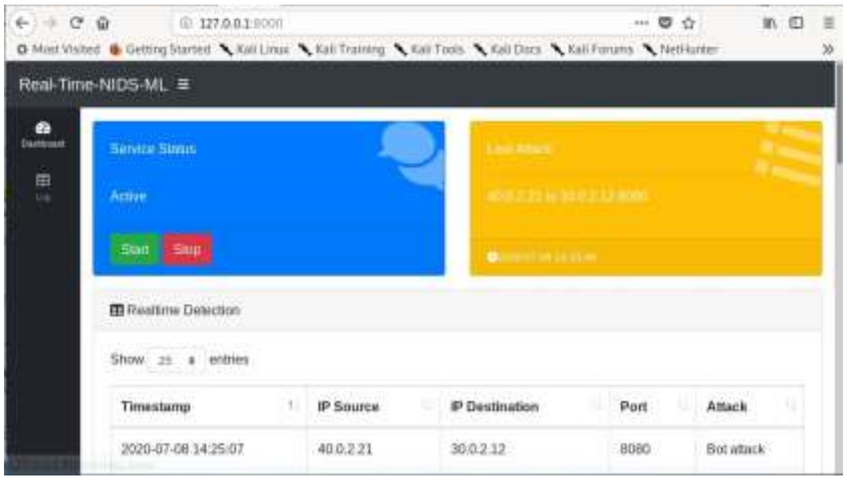


Fig. 6. *Design of the presented NIDS*

4.5 Experiment Setup

To make the presented NIDS user friendly, we developed a web application by using python 3 with Django framework. Fig. 7 shows the user interface of the presented NISD. We also used Google Colab environment [38] for conducting experiments with deep learning. Table 8 shows all packages used for implemented the presented NIDS.



Show 25 entries

Timestamp	IP Source	IP Destination	Port	Attack
2020-07-08 14:25:07	40.0.2.21	30.0.2.12	8080	Bot attack

Fig. 7. *User interface of the presented NIDS*

Table 8: Summary of packages used for implementing the presented NIDS

Libraries / Packages	**Function**
Numpy and Pandas	Used for data manipulation
Scikit_learn	Python module for machine learning
Channels	Used to handle WebSockets, chat protocols and HTTP request
Asgiref	a asynchronous web apps and servers to communicate with each other
Django	high-level Python Web framework
Requests	Making HTTP requests in Python
channels_redis	Channel layer that uses Redis as its backing store, and supports both a single-server and sharded configurations

We also used Redis-server for caching engine as the backing store for the channel layer. The Consumer will use a WebSocket (via Django Channels) that provides full-duplex communication. Whenever a user is authenticated, an event will be broadcasted to every other connected user. Each user's screen will be automatically updated with the latest data.

Moreover, we tested the presented NIDS by simulating many attack scenarios. Some machines have been used to simulate the presented NIDS. Table 9 describes all machines used for archiving this task.

Table 9: Machines used for simulating the presented NIDS

Operation System	Type	RAM	Storage
Kali linux	Attacker	4G	60
Windows 7 (Client 1)	Victim	1.5G	30
Windows 7 (Client 2)	Victim	1.5G	30
Web server	Victim	2G	40
Kali linux	IDS System	4G	60
OWASP Machine (web server)	Victim	4G	30

5. Experiment Results

We conducted many experiments for evaluating performance of selected machine learning techniques. We merged all attack in a single CSV file and applied selected classifiers. The accuracies are 96.9742% with Decision Tree (DT), 28.483% with Gaussian Naïve Bayes (GB), 96.695% with Random Forest (RF), 96.429% with Multi-layer Perceptron (MLP) and 94.68% with Logistic Regression (LR). We also used Keras Tensorflow environment and reporting the accuracy with 96.39%. Fig. 8 shows confusion matrix with Decision Tree classifier. We also built a model for detecting each attack in the dataset by using different classifier algorithms. Fig. 9 displays confusion matrix for each evaluated classifier. We can note clearly that RF provides the best performance. More details are shown in Table 10 to support our findings.

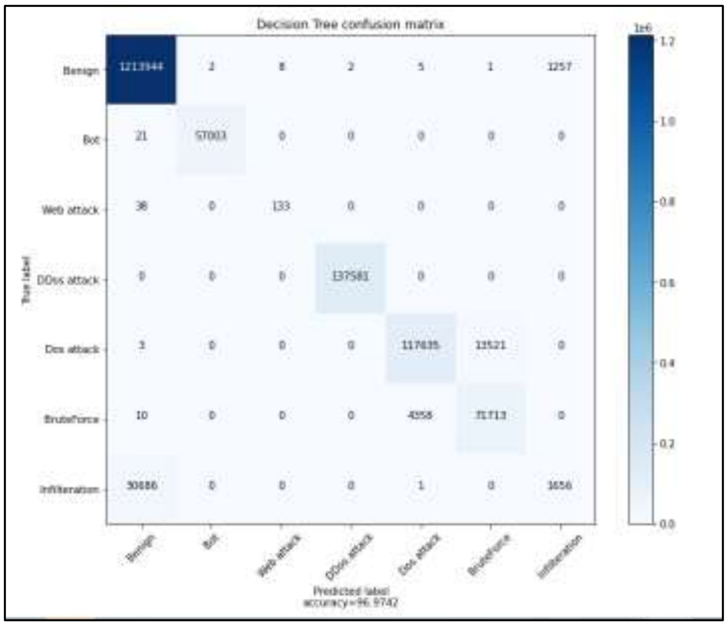


Fig. 8. Confusion matrix of evaluating Decision Tree classifier

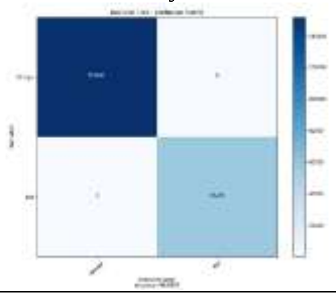
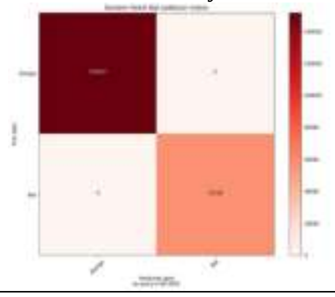
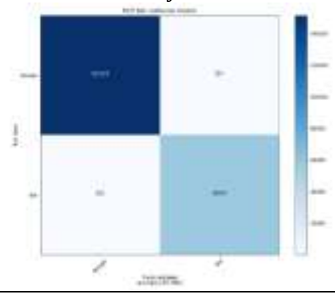
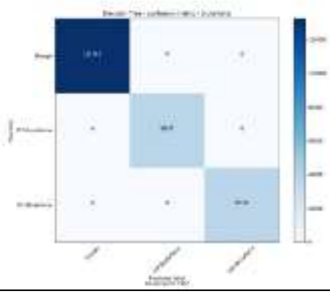
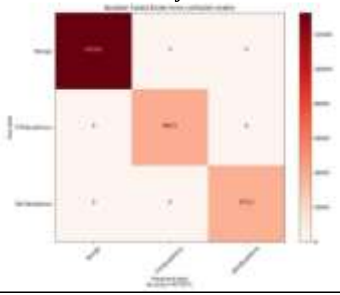
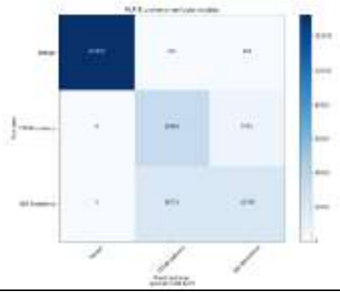
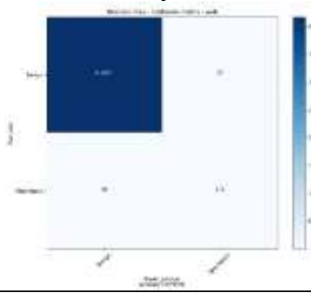
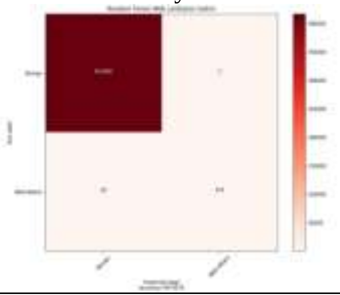
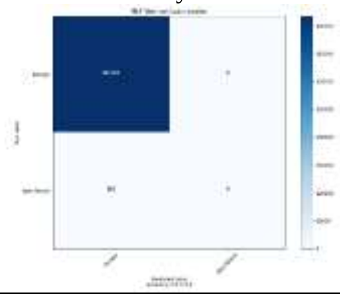
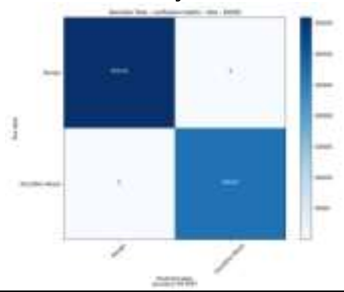
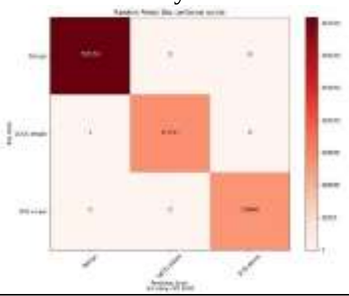
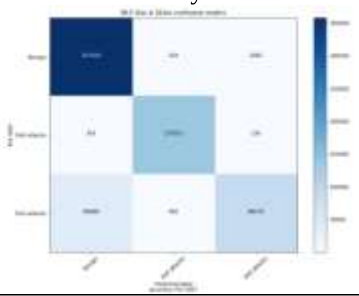
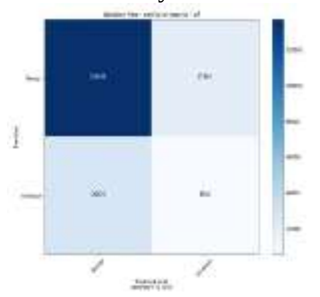
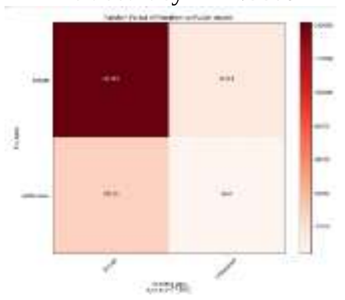
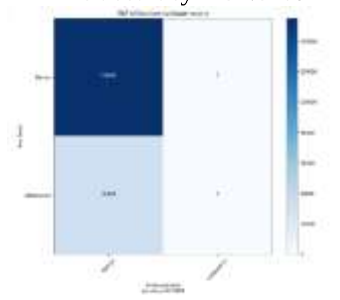
Mod./Att.	Decision Tree (DT)	Random Forest (RF)	Multi-layer Perceptron (MLP)
Bot Attack	The accuracy = 99.99% 	The accuracy=100% 	The accuracy = 99.78% 
Brute Force Attack	The accuracy = 99.99% 	The accuracy = 99.997% 	The accuracy = 88.125% 
Web Attack	The accuracy = 99.992% 	The accuracy = 99.997% 	The accuracy = 99.95% 
Dos &DDos Attack	The accuracy = 99.999% 	The accuracy = 99.999% 	The accuracy = 93.10% 
Infiltration Attack	The accuracy = 76.26% 	The accuracy = 77.88% 	The accuracy = 82.82% 

Fig. 9. Confusion matrices for evaluation detecting each attach with one classifier

Table 10: Evaluation of machine learning techniques

Classifier		**RF**	**M-NB**	**LGR**	**MLP**	**DT**
Accuracy		96.695	28.483	94.86	96.429	96.974
F1-score	Benign	0.99	0.12	0.97	0.99	0.99
	Bot	1.00	0.64	0.66	1.00	1.00
	Web attack	0.62	0.00	0.21	0.35	0.86
	DDos attack	1.00	0.77	0.99	1.00	1.00
	Dos attack	0.93	0.74	0.92	0.91	0.93
	Brute force	0.89	0.84	0.87	0.81	0.89
	infiltration	0.11	0.05	0.01	0.08	0.10
recall	Benign	0.99	0.06	1.00	1.00	1.00
	Bot	1.00	1.00	0.50	1.00	1.00
	Web attack	0.44	0.53	0.12	0.21	0.79
	DDos attack	1.00	1.00	1.00	1.00	1.00
	Dos attack	0.90	0.77	0.90	0.98	0.90
	Brute force	0.94	1.00	0.89	0.71	0.94
	infiltration	0.07	0.70	0.01	0.05	0.06
precision	Benign	0.98	0.98	0.95	0.97	0.98
	Bot	1.00	0.47	0.96	0.99	1.00
	Web attack	1.00	0.00	0.89	0.98	0.95
	DDos attack	1.00	0.63	0.97	1.00	1.00
	Dos attack	0.96	0.72	0.93	0.85	0.96
	Brute force	0.84	0.72	0.84	0.95	0.84
	infiltration	0.26	0.03	0.33	0.44	0.61

6. Conclusion and Future Work

In this work, we designed and implemented a Network Intrusion Detection System (NIDS) by using machine learning. This presented system detects attack in real-time. The system detects attack by using the model that was selected from different machine learning techniques. The model was selected by evaluating five classification algorithms: Decision Tree (DT), Random Forest (RF), Naive Base, Multi-Layer Perceptron (MLP) and Logistic Regression (LR). We evaluated selected algorithms by finding the best accuracy. The models was trained by using CSE-CIC-IDS2018 dataset which contains different types of attacks. We also developed a web application to make this system easy to use. The web application provides alerts when attacks happen and generate a log file for reporting all intrusions.

This work can be extended by improving performance of the presented system. One of the improving directions is based on employing more machine learning techniques for developing IDS. It is interested to evaluate performance of applying semi-supervised classification to the used dataset. We can also expand the system by enabling detection of attacks that affect host devices such as viruses, Trojan, malware, etc. Moreover, extracting additional features from the used dataset may improve the performance.

7. References

[1] Network Design: Firewall, IDS/IPS (https://bit.ly/344unu9), [Accessed: February 17, 2022]
[2] Important Model Evaluation Metrics for Machine Learning (https://bit.ly/2Tb49US),[Accessed : 1-Apr-2020]
[3] Basnet, R., Shash, R., Johnson, C.., Walgren, L., & Doleck, T. (2019). Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks, Journal of Internet Services and Information Security (JISIS), 9(4), pp. 1-17.
[4] Nagar, R., & Y. Singh (2019). A literature survey on Machine Learning Algorithms, Journal of Emerging Technologies and Innovative Research (JETIR), 6(4), pp. 471-474.
[5] Loganathan, G., Samarabandu, J., & Wang, X. (2018). Sequence to Sequence Pattern Learning Algorithm for Real-time Anomaly Detection in Network Traffic, IEEE Canadian Conference on Electrical & Computer Engineering (CCECE).
[6] Ahmim, A., Maglaras, L., Ferrag, M., Derdour, M., & Janicke, H. (2018). A Novel Hierarchical Intrusion Detection System based on Decision Tree and Rules-based Models, 15th International Conference on Distributed Computing in Sensor Systems
[7] Shone, N., Ngọc, T. N., Phai, V. D., & Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection, IEEE Transactions on Emerging Topics in Computational Intelligence.
[8] Van, N., Thinh, T., Sach, L. (2017). An anomaly-based Network Intrusion Detection System using Deep learning, International Conference on System Science and Engineering (ICSSE), pp. 210-214.

[9] Lin, P., Ye, K., & Xu, C. (2019). Dynamic Network Anomaly Detection System by using Deep Learning Techniques, Cloud Computing, CLOUD, pp.161-176.
[10] Abudalfa, S., & Mikki, M. (2013). A Dynamic Linkage Clustering using KD-Tree. International Arab Journal of Information Technology, 10(3).
[11] Abudalfa, S. & Mikki, M. (2013). K-Means Algorithm with a Novel Distance Measure, Turkish Journal of Electrical Engineering & Computer Sciences, 21(6), pp. 1665-684.
[12] Jain, M., Kaur, G. (2019). A Novel Distributed Semi-Supervised Approach for Detection of Network Based Attacks, 9th International Conference on Cloud Computing, Data Science & Engineering.
[13] Ghanem, K., Aparicio-Navarro, F. J., Kyriakopoulos, K., Lambotharan, S., & Chambers, J. (2017). Support Vector Machine for Network Intrusion and Cyber-Attack Detection, Sensor Signal Processing for Defence Conference (SSPD).
[14] Hijazi, A., El Safadi, E. & Flaus, J. (2018). A Deep Learning Approach for Intrusion Detection System in Industry Network, BDCSIntell.
[15] Sofi, I., Mahajan, A., & Mansotra, V. (2017). Machine Learning Techniques used for the Detection and Analysis of Modern Types of DDoS Attacks, International Research Journal of Engineering and Technology (IRJET), 4(6), pp. 1085-1092.
[16] Mohammadpour, L., Ling, T., Liew, C., & Chong, C. Y. (2018). A convolutional neural network for network intrusion detection system, The Asian-Pacific Advanced Network, 46, pp. 50-55.
[17] Yin, C., Zhu,Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks, IEEE Access.
[18] Wang, L., & Jones, R. (2017). Big Data Analytics for Network Intrusion Detection: A Survey, International Journal of Networks and Communications, 7(1), pp. 24-31.
[19] Tang, Mhamdi, L., T., Mclernon, D., Zaidi, S., & Ghogho, M. (2016), Deep Learning Approach for Network Intrusion Detection in Software Defined Networking, International Conference on Wireless Networks and Mobile Communications.
[20] Zhou, Q., & Pezaros, D. (2019). Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection - An Analysis on CIC-AWS-2018 dataset.
[21] Jamadar, R. A. (2018). Network Intrusion Detection System Using Machine Learning, Indian Journal of Science & Technology, 11(48).
[22] Filho, F., Silveira, F., Junior, A., Vargas-Solar, G., & Silveira, L. (2019). Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning, Hindawi, Security and Communication Networks.
[23] Loganathan, G. (2018), Real-time Intrusion Detection using Multidimensional Sequence-to-Sequence Machine Learning and Adaptive Stream Processing.
[24] Snort, Documents(NIDS/IPS) (http://www.snort.org), [Accessed: 28- Nov- 2019]
[25] Solarwinds , (NIDS/HIDS) (https://www.solarwinds.com), [Accessed: 28- Nov- 2019]
[26] Suricata, Real time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM) (https://suricata-ids.org), [Accessed: 29- Nov- 2019]
[27] Sagan, real-time log analysis & correlation engine (https://quadrantsec.com/sagan_log_analysis_engine/), [Accessed: 30- Nov- 2019]
[28] Ossec, HIDS (https://www.ossec.net) , [Accessed: 2- Nov- 2019]
[29]ACARM-ng,(NIDS/HIDS and IPS) (https://www.acarm.wcss.wroc.pl), [Accessed: 3- Nov- 2019]
[30] Fail2Ban, (IDS/IPS) (https://www.fail2ban.org), [Accessed: 4- Nov- 2019]
[31] Samhna,(HIDS) (https://la-samhna.de/samhain) , [Accessed: 4- Nov- 2019]
[32] Zeek, Home(Network Security Monitor) (https://www.zeek.org/index.html),[Accessed: 5- Nov- 2019]
[33] Firestorm NIDS, (NIDS) (http://www.scaramanga.co.uk/firestorm/), [Accessed: 5- Nov- 2019]
[34] Chkrootkit, (http:// http://www.chkrootkit.org/), [Accessed: 24-Feb-2022]
[35] Tripwire, (https://tripwire.dhs.gov/), [Accessed: 24-Feb-2022]
[36] A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018) (https://registry.opendata.aws/cse-cic-ids2018/), [Accessed: 27-May-2020]
[37] CSE-CIC-IDS2018 on AWS (https://www.unb.ca/cic/datasets/ids-2018.html),[Accessed: 27-May-2020]
[38] What is Google colab (https://mc.ai/what-is-google-colab/) , [Accessed : 21-Jun-2020]