SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I3P118

Volume 11 Issue 3, 195-208, March 2024
© 2024 Seventh Sense Research Group®

Original Article

Implementation of Six Single Classifiers and Feature Selection for Performance Enhancement in Anomaly-Based Intrusion Detection

Abdisalam A. Mohamed[1], Ibraheem Shayea[2], Fadi Al-Turjman[3]

[1]*Hormuud University, Mogadishu, Somalia.*
[2]*Istanbul Technical University, Istanbul, Turkey.*
[3]*Near East University, Mersin, Turkey.*

[1]*Corresponding Author : aam@hu.edu.so*

Received: 17 January 2024 Revised: 19 February 2024 Accepted: 18 March 2024 Published: 31 March 2024

Abstract *- Attacks against information systems have been sharply increasing recently. Cyberattacks are becoming less detectable by the normal antiviruses and firewalls. Various security systems have been deployed to protect information systems; Network Intrusion Detection Systems (NIDS) are among the most widely used security systems in the networking industry. IDS can be an anomaly-based or signature-based system. Signature-based NIDSs are effective against known attacks but futile against zero-day attacks. To detect novel attack techniques, anomaly-based IDS has proven to be more useful than signature-based IDS. This study used six Machine Learning algorithms to detect network intrusion incidents. The CSE-CIC-IDS2018 dataset is employed to train and test the algorithms. The dataset is cleared of defects, and important features are selected using the Random Forest Regressor algorithm. A sample of the dataset with selected key features is applied to six machine learning algorithms (Gradient Boosting, AdaBoost, ID3, KNN, MLP, and Random Forest). Within a short period of time, the algorithms achieved the following F1-Scores: Gradient Boosting (0.95), AdaBoost (0.94), K-Nearest Neighbors (0.93), ID3 (0.93), Random Forest (0.93), and MLP (0.78).*

Keywords *- AdaBoost, CSE-CIC-IDS2018, Machine Learning, MLP Network Intrusion Detection, Random Forest.*

1. Introduction

According to the International Telecommunications Union (ITU), only 0.4% of the world population had internet access in 1995. Now, more than 53% of the people around the world are connected to the internet [1]. The internet is one of the most significant technological achievements the world has ever seen in the 21st century.

However, the proliferation of this technological advancement came with immense security problems. The industry of computing is exposed to an exponentially growing likelihood of unexpected downtime because of various cyber-attacks and breaches. These violations are from cybercriminals and online terrorist operations [2].

Tother with the rise of the Internet of Things (IoT), and 5G mobile network, more devices will become interconnected. Our internet dependency will rapidly increase in our daily lives, leading to new threats as well [3]. For any given communication to be secure, it must have the following characteristics: confidentiality, integrity, and availability. Network intrusion is any set of activities that aims to compromise the pillars of network security.

To monitor systems for suspicious activity or policy infringements, a device or software application called the Network Intrusion Detection System (NIDS) is used [4]. NIDSs can be of two kinds based on their detection techniques: Signature-based NIDSs (SNIDS) and Anomaly-based NIDSs (ANIDS).

Signature-based NIDSs detect attacks by searching for specific patterns in network traffic or predefined, well-known, suspicious instruction tactics used by attackers. Signature-based NIDS performs well in detecting any known attack. However, for unknown attacks or cases where an anomaly is present in the network traffic, the performance of Signature-based NIDS sharply decreases [4].

Anomaly-based NIDSs detect unusual network activities by examining the general patterns of network traffic flow without the need to inspect each packet. ANIDSs have been successful in detecting previously unseen (novel) attacks, making them more efficient against zero-day attacks (unknown and unaddressed vulnerability) [5]. With the reality of ever-changing attack scenarios, ANIDSs have become the most appropriate intrusion detection technology. Any given operational IDS must be able to conduct the following three steps: track and collect network flow data, clean raw data and convert it to an input format, and classify network traffic into normal or malicious traffic.

In many studies, machine learning algorithms have been used to enhance IDS performance, specifically in achieving the most accurate detection rate and lowest false positive rates. The quality of the dataset used to train machine learning models contributes to its effectiveness; however, publicly available datasets contain more records and features, which will contribute to a longer training and testing time, an issue most of the studies did not address adequately.

To bridge that gap, this study aims to improve time efficiency in the training and testing of machine learning algorithms used for intrusion-detecting purposes. To shorten the training and testing times of the algorithms, the most important features are selected, and using only a few of the most important features, six machine learning models are trained and tested. The study achieved substantial improvement in detecting intrusion fast.

Another obstacle that the studies on intrusion detection face is the use of an old dataset; new attack types have emerged, and the old datasets do not reflect well on the newer attack types. This paper presents an anomaly-based network intrusion detection technique using machine learning algorithms trained with a sample of a benchmark dataset (CSE-CIC-IDS2018) with its features greatly reduced. The goal of this study is to contribute to the time efficiency enhancement of machine learning algorithms in the intrusion detection domain.

The rest of the paper is organized as follows. Section 2 explains the characteristics of the dataset used. Section 3 reviews several previous research on NIDS. Sections 4 and 5 discuss the details of the implementation method as well as the results, respectively. Section 6 concludes this paper lastly.

2. The Dataset

The CSE-CIC-IDS2018 dataset was developed by Canada’s national cryptologic agency called the Communications Security Establishment (CSE), together with the Canadian Institute for Cybersecurity (CIC). It contains fourteen distinct attacks grouped under six separate attack cases: DoS, DDoS, Brute-force, Infiltration, Web attacks, and Botnet. During the dataset development phase, an attacking infrastructure and a victim organization were developed. The attacking side had fifty machines, and the attacked organization being targeted had five departments with 30 servers and 420 machines in total.

Using the CICFlowMeter-V3 tool [6], eighty features were chosen to represent the characteristics of the traffic generated in the network (system logs and network traffic of all machines). The end product is a diverse, inclusive, and world-class dataset in the intrusion detection domain. It was made to reflect on complete user profiles to represent events and behaviors observed in the network.

2.1. FTP-Brute Force

According to Kaspersky [7], a brute force attack aims to break a password or username, discover a web page, or obtain the encryption key of a message [7]. It ultimately relies on trial and error. Depending on the complexity, the length of the victim’s password or username, and the technical level of the attacker, a brute-force attack may take anywhere from milliseconds to years. It is one of the most popular attack methods that hackers use despite being an old technique that existed before the internet in the form of a cryptanalytic attack [8].

Metasploit modules, Hydra, Ncrack, Nmap NSE scripts, and Medusa are some of the most popular tools used for password-cracking and brute-force attacks. Other tools, such as Hashpump and Hashcat, are used in password hash cracking. To create the CSE-CIC-IDS2018 dataset, Patator was utilized since it is among the most complete multithreaded, reliable, and flexible tools written in Python. Two modules of brute-force attacks are present in this dataset: FTP and SSH. A Kali Linux machine was set to be the machine to execute the attack and a machine running Ubuntu 14.0 operating system was the target machine [8].

FTP is a network protocol intended to facilitate the transfer of files between client and server in a network. For any file transfer to occur through FTP, verified credentials (username and password) are mandatory. The FTP-Patator attack aims to illegally acquire this username and password, thus compromising the victim’s system [8].

2.2. SSH-Brute Force

SSH is a network security protocol that encrypts and decrypts the operations of various services throughout the network. SSH had overtaken Telnet (which is not encrypted) in remote access aspects. That popularity makes SSH a vital target for cybercriminals, especially SSH Brute-force attacks [9,10].

SSH brute force is one of the most popular forms of attacks waged to access a remote machine by performing repetitive authentication credential guessing trails using all possible password combinations until the correct one is reached [11]. The SSH-Brute-force data in the CSE-CIC-IDS2018 dataset was created using the same Patator tool [12] used in the FTP Brute-force attack.

2.3. Brute Force-XSS

Cross Site Scripting attacks are in the list of well-known web attack techniques. It all begins with the injection of a malicious piece of code into the targeted webpage. When a legitimate user visits this webpage, the code is immediately executed without being noticed. The results can be very devastating to the extent that they may include identity theft, impersonation, and arbitrary code execution [13].

2.4. Botnet

The word 'Bot' originates from the word "Robot." It is a common term for describing an automated process, such as a Google bot, that collects information from the internet. Bots were created with good intentions; however, they can also be used for malicious activities (such as information theft) or as launching pads for distributed attacks. Malicious software is stealthily installed on the targeted machine to infect it. Control of the machine is then passed to a remote attacker, who will then have direct command. Infected machines taken hold by attackers are referred to as zombie machines. They are primarily used to wage Distributed Denial of Service attacks (DDoS) [14].

Botnets use a level command and control mechanism with a wide range of protocols. Such diversity makes botnet detection difficult. During passive mode, limited or no activities are present in the network, making Botnet detection even more challenging [15]. In the active mode, the packet flow and packet size with the TCP Push (PSH) flag used to speed up packet flow help IDSs detect Botnets [16].

A Trojan horse malware package was applied for the CSE-CIC-IDS2018 dataset. It runs on the version of Microsoft Windows called Zeus, used to simulate many malicious and criminal acts such as banking information theft by form grabbing and man-in-the-browser keystroke logging. It can also be used to install Crypto-Locker ransomware. During dataset generation, an open-source botnet (called Ares botnet) was utilized. Ares has the following abilities: File upload/download, Keylogging and Remote cmd.exe shell Persistence Screenshot. During the execution scenario, target machines were infected with Zeus and Ares botnets, and screenshots were requested from the zombies every 400 seconds [17].

2.5. DOS-GOLDEN-EYE

DoS attacks directly aim at blocking legitimate users' rights to access the system by reducing system availability. DoS attacks flood networks with malicious loads consisting of superfluous network datagrams or normal packets that fill more CPU processing capabilities, network buffers, and overall network bandwidth. As system resources form a bottleneck, performance drops, and the targeted system may eventually crash [18]. Although DoS attacks consist of various types, four DoS attack types were applied on the CSE-CIC-IDS2018 dataset: DoS-Hulk, DoS-GoldenEye, DoS-SlowHTTPTest, and DoS-Slowloris. GoldenEye is a multithreaded Python-based HTTP DoS Tool used to test site vulnerabilities during DoS attacks. It allows numerous parallel connections to a given URL. It applies the Keep-Alive method to substantially increase the size of files transmitted over a given TCP connection. It also deactivates HTTP-Cache-Control using the NoCache message feature. By using an attack vector containing 'HTTP Keep-Alive and NoCache', GoldenEye's attack can immediately deplete system resources. Attack packets of this attack type are unencrypted. They do not support fabricated IP (spoofed) addresses [19].

2.6. DOS-HULK

HULK, short for HTTP Unbearable Load King, is a multi-purpose attack tool that can be utilized to launch both DDoS and DoS attacks. It can take down servers in a very short duration of time by generating HTTP GET flood requests and TCP SYN flood. Another extremely dangerous trait is that it can completely obscure the real user platform and instead employ a dissimilar sample for separate attack execution [20].

2.7. DOS-SLOWLORIS

Slowloris is a tool developed using Perl programming language where both the GUI and command line interfaces are used to generate an attack. It floods the victim with TCP SYN requests to the target machine. When this type of attack targets a web server, incomplete and smaller-sized TCP packets with SYN flags can be observed [20].

2.8. DOS-SLOWHTTPTEST

Slow Read DoS attack was created by Sergey Shekyan [21]. In this scenario, a hacker typically sends a legitimate HTTP request to the victim's Web server and then reads the response very slowly. The attacker sends a legitimate request after the normal TCP three-way handshake. Next, the attacker advertises a window size smaller than the average size and slows down the HTTP response operation.

Upon advertising a window size of 0, the Web server stops sending data despite maintaining an open connection. Receiving more authentic requests makes the Web server quickly reach maximum capacity and stop serving legitimate clients. Since this attack uses legitimate HTTP requests, it is extremely difficult to segregate normal traffic from malicious ones. To safeguard networks against this type of attack, the network layer must be continuously monitored with more focus on small-sized packets [20].

2.9. DDOS-LOIC (UDP/HTTP)

DDoS is an organized attack directed to stop authentic users from accessing network resources, even causing systems to crash from extraordinary load. It can be executed with a massive number of compromised hosts. At the preliminary stage, attackers spot vulnerabilities in a given network and infect a vast number of machines with malware to remotely control them eventually. In the second phase, attackers exploit the infected machines (Zombies) to spread malicious packets to the attacked machine(s) without the infected machines realizing their participation in these harmful activities. Depending on the concentration of attack packets and the number of zombies used to launch the attack, subsequent damage will be inflicted on the victim's network and machines [22].

LOIC is a free network testing platform created by Praetox technologies for the network stress test. It is a Graphic User Interface-based DDoS attack platform that can run on Windows and macOS operating systems. Its ease of use and widespread availability allow criminals to efficiently conduct coordinated and severe DDoS attacks without the need for more knowledge or experience [23].

2.10. DDOS-HOIC

High Orbit Ion Cannon (HOIC) is a free, BASIC-written, speedy, multithreaded, network stress testing tool that can take down hundreds of websites at once through HTTP flood and POST requests sent to the target server. HOIC has the ability to drain the resources of the victim's server [24].

2.11. SQL Injection

Uncontrolled input fields at the user interface of applications with bugs help attackers alter the SQL commands and queries directed to the database. Such attacks are known as SQL injection attacks. These attacks are prevalent since finding vulnerabilities and exploiting them is extremely easy.

The damages inflicted on the victim are phenomenal due to direct database access. A successfully executed SQL injection attack provides the hacker with unauthorized executive command over data, access to privileged database accounts, imitate a legitimate user, and access the web server [25].

2.12. Infiltration

Network infiltration is a network attack from the inside. It is mainly accomplished by exploiting vulnerable software such as Adobe Acrobat Reader. After successfully completing the exploitation phase, a backdoor is set up on the targeted machine to allow different attacks to be waged on the victim's network (such as IP sweep, service enumerations using Nmap, and port scan) [26].

3. Related Work

In recent years, numerous research that apply machine learning in the intrusion detection domain have been published. This section summarizes the key discoveries. In 2005, Chebrolu et al. [27] defined computationally useful features for IDS applications using two popular algorithms: Classification and Regression Trees (CART) and Bayesian Networks (BN).

The researchers established a hybrid model of ensemble classifiers for intrusion detection purposes. After training the model on the KDD CUP99 dataset, the study achieved an impressive 100% accuracy for Normal (Benign) traffic, Probe, and DOS attack detection. They also acquired an average accuracy of 84% for detecting U2R and R2L scenarios.

Weiming Hu et al. [28] presented an AdaBoost algorithm-based IDS using decision stumps as weak classifiers. The weak classifiers are simultaneously applied with continuous and categorical features, forming a strong classifier. An adaptable initial weights strategy was employed for minimum overfitting, leading to enhanced performance.

From the KDD CUP99 dataset, four attack types were examined in the experiments with and without handling overfitting techniques: DOS, U2R, R2L, and Probe. A 99.159% detection rate was accomplished without handling overfitting, and a 99.166% accuracy was achieved with handling overfitting.

The researchers proposed that their algorithm has a lower error rate and computational complexity in comparison to other algorithms tested on the same dataset. The aforementioned studies presented Machine Learning models trained on old intrusion detection datasets, making the detection of new attack types challenging; in this study, however, the CSE-CIC-IDS 2018 dataset, which is a newer and more diverse dataset, is used to train the machine learning models.

In 2013, Warusia Yassin et al. [29] created a new model that combines the Naïve Bayes Classifier and K-Means Clustering algorithms to prevent false alarm constraints. The IDS model was implemented on the ISCX 2012 dataset. The researchers accomplished a high-performing IDS of 99% accuracy using their algorithm combination and a 98.8% accuracy using Naïve Bayes alone.

In 2017, Sharafaldin et al. [26] implemented seven popular machine learning algorithms on the CICIDS2017 dataset to detect more than thirteen attack types: Multilayer perceptron, AdaBoost, Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors (KNN), Random Forest (RF), ID3 and Naïve-Bayes (NB). The Random Forest Regressor was used to operate with the minimum but the most important features. The study achieved the following results for each algorithm: ID3 (98%), KNN (96%), RF (97%), Naïve-Bayes (84%), AdaBoost (77%), MLP (76%), and QDA (92%).

Kanimozhi and Jacob [30] proposed an IDS model for Botnet attack detection that targets the financial sector. The IDS model was created using artificial intelligence trained with the IDS2018 dataset. The model developed in this paper is presented as a real-life applicable system with a superb 99.97% accuracy, 0.999 average area under the ROC curve, and a minimal false-positive rate of 0.03. To ensure an even more powerful, the study suggested the use of a GPU-based framework instead of a CPU-based one.

Zhou & Pezaros [31] implemented six algorithms (MPL, QDA, Random Forest, ID3, Naïve Bayes and KNN) with the CSE-CICIDS2018 dataset to protect against Zero-Day attacks. In the training phase, the models were trained with fourteen attack scenarios in the dataset. During the testing phase, eight novel (Zero-Day) attack types and regular traffic were used. The model accuracy was evaluated under the following parameters: Recall, Precision, F1-Score, and time overhead. Decision tree algorithms outperformed all other algorithms. The model has 100% accuracy on Zero-Day attacks and normal data only and 96% accuracy for scrambled Zero-Day attacks and benign data, with 5% false-positive rates.

Lin et al. [32] presented a robust anomaly detection system using deep learning methods. The study adopted Long Short-Term Memory (LSTM) to create a deep neural network architecture. An Attention Mechanism (AM) was also employed to improve the output of the model. To overcome the class imbalance issue in the CSE-CICIDS2018 dataset, SMOTE and an improved loss function algorithm were deployed. The model achieved a 96.2% accuracy.

D'hooge et al. [34] investigated the efficiency of supervised machine learning algorithms trained on separate normal traffic from various types of attack traffic. Botnet and DoS/SSL attack types, represented in two popular datasets (CIC-IDS2017 and CSE-CIC-IDS2018), were used to train and test twelve supervised learning algorithms. The credibility of inter-dataset generalization of the trained models was also examined. The study saw reduced effectiveness of machine learning algorithms used for intrusion detection systems with dataset(s) other than those they were trained with. Therefore, further experimentation on the same domain was suggested.

Fitni & Ramli [35] suggested an ensemble learning method which combines the advantages of individual detection algorithms. The study compared seven classifiers and presented the best suitable basic algorithms for ensemble learning. Logistics regression, Gradient Boosting, and Decision trees were selected for the ensemble model, which was later trained and tested with the IDS2018 dataset. For the feature engineering steps, Spearman's rank correlation coefficient was used for important feature identification. With the reduced number of features (23 of 80 features), the model achieved 98.8% precision, 98.8% accuracy, 97.9% F1-Score and 97.1% recall.

Karatas et al. [36] used Gradient Boosting, RF, *k*-NN, Adaboost, Linear Discriminant Analysis, and DT to differentiate attack traffic and benign traffic in the CICIDS2018 dataset. The Synthetic Minority Oversampling Technique (SMOTE) technique was applied to the dataset to solve class imbalance. The models were built on a Python environment utilizing Scikit-learn TensorFlow, Keras, and. The dataset was preprocessed to tackle the computational problems arising from some empty values and "Infinity". One-hot encoding was also employed. Five-fold cross-validation was used with 80% of the samples in the training session. The other 20% of instances were used as a test dataset; after implementing SMOTE, the overall size of the dataset expanded by around 17%. The study found that Adaboost was found as the best-performing algorithm, with an average precision score of 99.70%, an accuracy level of 99.69%, and a recall score of 99.69%.

Li et al. [37] employed feature selection and clustering to the CSECIC-IDS2018 dataset, including live real-time detection using an autoencoder classifier. In the preprocessing phase, "Infinity" and "NaN" instances were converted to 0. The dataset was first grouped into sparse and dense matrices and then normalized using L2 regularization.

The model was developed using Python, and important features were chosen using the Random Forest algorithm. 85% of the dataset was applied for training, while 15% was used for testing. The popular Affinity Propagation clustering algorithm was implemented on 25% of the training data to collect features into subgroups and, in turn, fed to the autoencoder. The recall rates of the developed model and that of another autoencoder model (named Kitnet) were compared. The two compared models achieved a recall of 100% for several attack types. The researchers proposed that their proposed model achieved a faster detection than that of KitNet.

Ramos et al. [38] assessed a group of five learners using the CSE-CIC-IDS2018 dataset with the ISOT HTTP Botnet [39], a Botnet dataset containing normal and malicious records of DNS traffic. Several machine learning algorithms were used, including Random Forest, Decision Tree, Naive Bayes, k-Nearest Neighbor (*k*-NN) [31], and SVM. Different Feature engineering techniques with the feature importance method of Random Forest were used. Nineteen important features were selected for the IDS2018 dataset, while twenty features were chosen for the ISOT HTTP dataset.

The models were developed using Scikit-learn. 80% of the sample was used for training and 20% for testing. A five-fold cross-validation was subsequently implemented, and the Grid Search technique was deployed for optimization purposes. For the CSE-CIC-IDS2018 dataset, the Random Forest and Decision Tree algorithms scored 99.99% accuracy, 100% precision, and 99.99% recall. Similarly, Decision Tree and Random Forest learners achieved the best accuracy for ISOT HTTP.

Zuech et al. [40] highlighted the effects of class imbalance in machine learning applications regarding cybersecurity. Seven classifiers were trained and tested on the CSE-CIC-IDS2018 dataset to detect web attacks and normal traffic: CatBoost (CB), Decision Tree (DT), XGBoost (XGB), Random Forest (RF), Naive Bayes (NB), Logistic Regression

(LR) and LightGBM (LGB). Several Random Undersampling (RUS) steps were made.

Both the Area Under the Precision-Recall Curve (AUPRC) and the Area Under the Receiver Operating Characteristic Curve (AUC) were used as performance matrices. The study found that different random undersampling ratios statistically lead to varied performances in separating web attacks from normal traffic. Distinct classifiers also statistically achieve different accuracy levels.

Most of the previous IDS papers had little intent in reducing the number of features in the dataset, Therefore, reducing the number of dataset features to a few but important features should be considered as an important factor.

4. Methodology

Several pre-processing steps are made to clean the dataset from the defects observed before implementing machine learning algorithms that classify benign and malicious traffic. A cleaned sample of the dataset is selected and divided into two parts: a set for training and a set for testing. Since the dataset contains 80 features, the most important features are selected to be applied to the algorithms. The experiments in this study have been conducted in two approaches: single specific attack type detection and overall attack and benign classification. Figure 1 illustrates the implementation process.

Machine learning experiments are implemented on a machine with 8GB RAM and Intel (R) Core (TM) i7-4712HQ CPU @ 2.30 GHz using the Microsoft Windows 10 operating system. We used the Sklearn module in Python 3.8 to implement the algorithms. Six machine learning algorithms are implemented: Adaboost, Decision Tree (ID3), Random Forest, Gradient Boosting, K-Nearest Neighbors (KNN), and Multilayer Perceptron (MLP) [41-43].

Table 1. ML Algorithms hyperparameter values

ML Algorithms	Hyperparameters
Adaboost	Estimator = 50, Learning Rate =1.0, Algorithm = “samme R”
Random Forest	Minimum Samples split =2, minimum samples leaf =1
Decision Tree	Sample split =2, Minimum samples leaf =1, Estimator = “warn”, Criterion = “Gini”
Gradient Boosting	Estimator = 100, Maxi Depth =3, Validation =0.1
MLP	Hidden Layer size =(13,13,13), Max inter =500
KNN	Class =5, Weight = “Uniform”, Distance = “Minkowski”

4.1. Dataset Cleaning

The IDS 2018 dataset contains network traffic in Packet Capture (Pcap) format, logs, and ten pre-processed and labelled CSV files. The CSV files contain more than 16 million records. Only a sample of those records (around 9 million) were used in this research.

Defects in the selected dataset that require cleaning have been addressed. Missing and infinite values were replaced with 0 and 1, respectively. Records with more redundant or missing values were deleted. Benign and malicious labels were encoded with 0 and 1, respectively. To address the class imbalance in the original dataset, samples representing each attack type together with normal traffic samples were collected in a separate file with a 30:70 ratio.

Similarly, a single file containing all attack types represents an attack, and benign samples represent the normal attack dataset. The two dataset groups were utilized to train and test the models in different experiments. Table 3 illustrates the samples of each attack for the dataset utilized in this study.

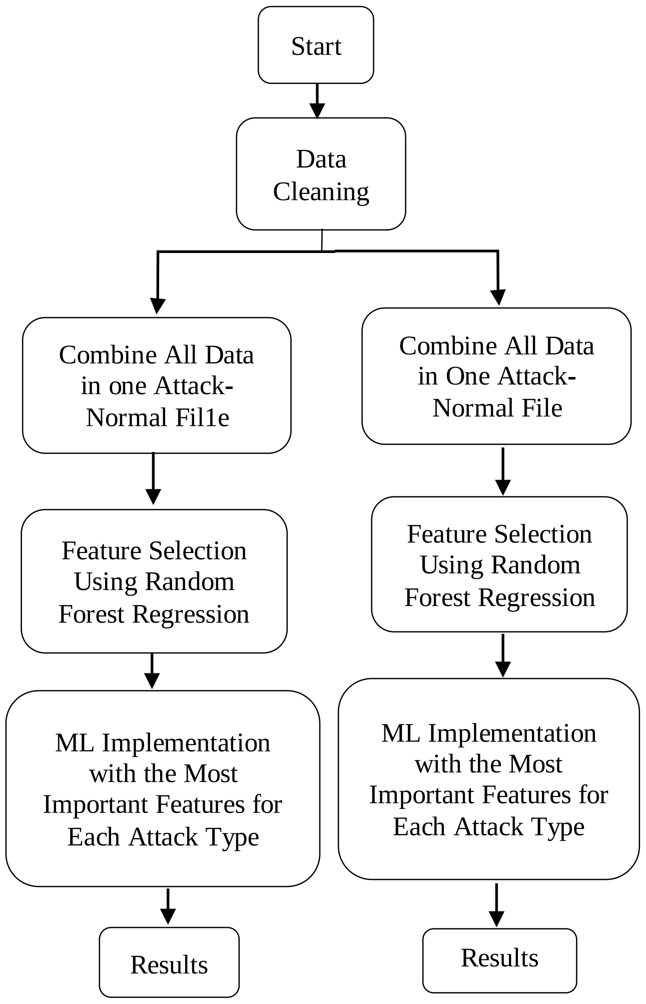
Start
Data Cleaning
Combine All Data in one Attack-Normal Fil1e
Combine All Data in One Attack-Normal File
Feature Selection Using Random Forest Regression
Feature Selection Using Random Forest Regression
ML Implementation with the Most Important Features for Each Attack Type
ML Implementation with the Most Important Features for Each Attack Type
Results
Results

Fig. 1 Experimental setup

4.2. Feature Selection

Training machine learning algorithms with a dataset that has many features can cause multiple problems, such as overfitting for instance. It may result in an apparent increase in the accuracy of the model at the training stage but degrades the effectiveness of the model on previously unseen data in the testing phase. It may also extend the training time of the algorithms. Selecting a descriptive subset of features is a good method to avoid overfitting and simultaneously improve the model's accuracy when fed with new data. The use of a smaller number of features substantially reduces the computational cost of training and prediction as well [44]. This study utilized the Random Forest Regressor algorithm, which is based on the tree concept for the feature selection procedure. This algorithm enables features that contribute even a limited number of trees to remain visible. In the implementation phase, the Random Forest Regressor class of the Sklearn library is used to calculate feature importance [45].

Table 2. Important features selected for each attack type

Attack Type	Features	Attack Type	Features
Bot	Bwd Pkt Len Mean	**DoS Attacks-Hulk**	Flow IAT Min
	Flow Pkts/s		Bwd Pkt Len Std
	Flow IAT Mean		Fwd Pkt Len Max
	Flow IAT Min		Fwd Pkt Len Std
	Fwd Pkt Len Mean		Bwd Pkt Len Max
Brute Force -Web	Flow Byts/s	**DoS Attacks-SlowHTTPTest**	Flow Pkts/s
	Fwd Pkt Len Mean		Flow IAT Mean
	Flow IAT Mean		Flow Duration
	Flow Duration		Flow IAT Max
	Flow IAT Max		Tot Fwd Pkts
Brute Force -XSS	Fwd Pkt Len Mean	**DoS attacks-Slowloris**	Fwd Pkt Len Max
	TotLen Fwd Pkts		Flow Byts/s
	Tot Bwd Pkts		Tot Bwd Pkts
	Fwd Pkt Len Max		Flow IAT Min
	Bwd Pkt Len Mean		Fwd IAT Tot
DDOS Attack-HOIC	Flow Duration	**FTP-BruteForce**	Flow IAT Mean
	Flow IAT Max		Flow Duration
	Bwd Pkt Len Std		Flow IAT Max
	Fwd IAT Tot		Tot Fwd Pkts
	Tot Bwd Pkts		Flow IAT Min
DDOS Attack-LOIC-UDP	TotLen Fwd Pkts	**Infiltration**	Flow Byts/s
	Flow Duration		Flow IAT Min
	Bwd Pkt Len Mean		Flow IAT Max
	Flow IAT Min		Flow Duration
	Flow IAT Max		Flow IAT Mean
DDoS Attacks-LOIC-HTTP	Flow IAT Max	**SQL Injection**	Flow IAT Max
	TotLen Fwd Pkts		Flow Byts/s
	Flow Duration		Bwd Pkt Len Max
	Flow IAT Std		Bwd Pkt Len Std
	Fwd Pkt Len Std		Flow Duration
DoS Attacks-GoldenEye	Fwd Pkt Len Max	**SSH-Bruteforce**	Flow IAT Mean
	Flow Duration		Flow Pkts/s
	Fwd IAT Tot		Flow Duration
	Flow IAT Max		Flow IAT Max
	Fwd Pkt Len Mean		Flow IAT Min

The extensive need for domain knowledge in feature engineering widely exists for network intrusion detection. Several key considerations must be made when training a model with a given dataset. Numerous features can be misleading. They may look significant; however, their underlying importance in identifying network anomalies can be relatively small or non-existent.

Table 3. Number of records of each class in the cleaned dataset

Traffic Type	Number of Samples
Benign	6584535
DDOS attack-HOIC	686012
DDoS attacks-LOIC-HTTP	576191
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infiltration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87
Total Number Samples	9332770
Total Attack Samples	2748235 (28.9%)
Total Benign Samples	6584535 (71.1%)

Packet inspection and network flow monitoring are examples of two popular network traffic monitoring methods. In a typical network-based anomaly detection process, the traffic traversing the network should be monitored for suspicious activity. The detection mechanism should also be fast and efficient. To meet these demands, flow-based analysis has become the preferred option since it allows attack detection through packet header information instead of packet payload information, as applied in packet inspection.

The five most popular attributes used to define a network flow are Destination IP, Source IP, Destination Port, Source Port, and Network Protocol [8]. Attackers are noticeably evading these well-monitored features since more attention is placed on such attributes. They avoid using well-known ports and sidestep security apparatus by using generated/fake IP addresses (spoofing). Port numbers can also range from 1 to 65535, indicating that port numbers other than those observed in training sessions may appear in the testing phase, thus disrupting the evaluation mechanism. Some applications are transmitted over the same port as well. Focusing on port numbers or any other feature related to source or destination ID will, therefore, be terribly misleading [46, 10]. Table 2 displays the five most important features of each attack type.

4.3. Evaluation Matrices

In the case of attack and benign classifications, the following four cases can occur: False Positive (FP): a benign sample classified as an attack sample; True Positive (TP): an attack sample classified as a malicious sample; False Negative (FN): an attack sample classified as a benign sample; True Negative (TN): a benign sample classified as a benign sample.

- Accuracy: The ratio of records correctly classified to the total number of samples.

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

- Recall: The ratio of correctly classified attack samples to the total number of attack samples.

$$Recall = \frac{TP}{FN+TP} \quad (2)$$

- Precision: The ratio of correctly classified attack samples to the total number of samples classified as an attack.

$$Precision = \frac{TP}{FP+TP} \quad (3)$$

- F1-Score: weighted average of precision and recall.

$$F1 - score = 2 \times \frac{Precission \times Recall}{Precission + Recall} \quad (4)$$

5. Results

This section illustrates the outcomes of the machine learning experiments. The implementation is conducted in two phases: individual attack classification and overall malicious-normal classification. The outcomes are evaluated in terms of the running time, precision, recall, F1 scores, and accuracy levels of ML algorithms.

5.1. Individual Attack Detection

The five features with the highest level of importance in the dataset for each attack are computed during the feature selection process, as indicated in Table 2. Machine learning algorithms are trained and tested using only these features to detect anomalies instead of training with all 80 dataset features. Table 4 presents the F1 scores for each algorithm. The F1-Score was specifically selected since it combines recall and precision. Therefore, it is the preferred evaluation matrix compared to other options. Most of the attacks are detected easily; Botnet and DDoS are two prominent examples of easily detected attack files, as shown in Tables 5 and 6.

In this study, all machine learning algorithms couldn't achieve substantial detection rates on Infiltration attacks, and the same is also observed in all previous works conducted on IDS 2017 and IDS 2018 datasets.

Table 4. F1-scores of classifiers for specific attack dataset

Attack Types	ML Algorithms					
	RF	KNN	ID3	AdB	MLP	GrB
	F1-Score	F1-Score	F1-Score	F1-Score	F1-Score	F1-Score
Brute Force -Web	0.91	0.92	0.89	0.92	0.72	0.9
Botnet	0.99	1.0	1.0	1.0	1.0	1.0
Brute Force -XSS	0.86	0.75	0.88	0.88	0.77	0.88
DDOS HOIC	0.98	0.99	0.98	0.98	0.93	0.98
DDOS LOIC-UDP	1.0	1.0	1.0	1.0	0.94	1.0
DDoS LOIC-HTTP	0.99	0.99	0.99	0.99	0.43	0.99
DoS GoldenEye	0.97	0.98	0.97	0.97	0.41	0.98
DoS Hulk	0.98	0.97	0.98	0.98	0.95	0.91
DoS Slow HTTPTest	0.98	0.98	0.98	0.98	0.94	0.97
DoS Slowloris	0.94	0.99	0.98	1.0	0.88	1.0
FTP-B-Force	0.98	0.98	0.98	0.98	0.98	0.98
SSH-B-Force	0.99	0.99	0.99	0.99	0.9	0.99
Infiltration	0.57	0.67	0.58	0.59	0.49	0.62
SQL Injection	0.88	0.82	0.81	0.89	0.6	0.9

Table 5. Botnet

ML Algorithm	Accuracy	Precision	Recall	Time (s)
Random Forest	1	1	1	0.047354
ID3	1	1	1	0.018748
AdaBoost	1	1	1	0.353962
MLP	0.99654	0.99460	0.9974	0.99602
KNN	0.99827	0.99728	0.9987	0.156244
GBoost	1	1	1	0.076038

Table 6. DDoS HOIC

ML Algorithm	Accuracy	Precision	Recall	Time (s)
Random Forest	0.9948	0.9927	0.9946	12.347
ID3	0.9953	0.9921	0.9963	3.6567
AdaBoost	0.9964	0.994	0.9961	74.740
MLP	0.9964	0.9951	0.9961	537.52
KNN	0.9971	0.9958	0.9970	75.725
Gboost	0.996	0.9955	0.9967	45.971

Table 7. Infiltration

ML Algorithms	Accuracy	Precision	Recall	Time (s)
Random Forest	0.7488	0.78155	0.5824	6.72903
ID3	0.74958	0.77807	0.5846	2.8575
AdaBoost	0.74694	0.73492	0.5923	40.8661
MLP	0.72507	0.77078	0.5353	99.290
KNN	0.73532	0.67620	0.6633	21.771
Gboost	0.75127	0.72498	0.6102	24.096

Tables 5 and 6 above show the accuracy, precision, and Recall together with running times of the machine learning algorithms on Botnet and DDoS HOIC attack types. The number of Botnet samples in the dataset seems to be sufficient for training models well; one positive thing that is observable from the results achieved by the classifiers is the running times of the algorithms; all six algorithms completed the training and detection process in record time, this will be an important factor in detecting real attacks swiftly.

Infiltration attacks are the hardest to detect, with a maximum F1-Score of 0.67. This was followed by SQL injection and Brute force-XSS attack types which are also classified with low F1-Scores of around 0.88. The latter two are underrepresented in the dataset, indicating that a few misclassified samples may significantly degrade the overall result. The infiltration attack samples in the dataset are more (161934 samples), yet they remain harder to detect.

According to the dataset's documentation [17], unrestricted file upload is recognized as an infiltration attack scenario. Traffic at the network level becomes difficult when distinguishing between illegitimate and legitimate file uploads. This attack type should be further examined to achieve a more desirable detection accuracy.

From the above tables, it is observable that MLP achieves the lowest performance level for all attack files and with longer running times than all other machine learning algorithms. In contrast, Random Forest and Decision Tree have achieved better performance than other methods in the shortest running time.

5.2. Attack-Normal Classification

In this section, the whole dataset is collected in a single file. Samples of all attack samples are labeled as "attack". Six machine learning algorithms are applied to this dataset. The features selected for individual attack types shown in Table 2 are combined, and sixteen unique features shown in Table 8 are used in this phase of the experiment.

Table 8. Sixteen important features were collected from all individual attack files

Features	Importance
Bwd Pkt Len Mean	1.939715e-01
Flow IAT Mean	4.710189e-02
Fwd Pkt Len Mean	1.051510e-02
Flow IAT Max	0.010301
Tot Bwd Pkts	0.028025
Flow Duration	0.100301
Fwd IAT Tot	0.000095
Fwd Pkt Len Std	2.588169e-03
Flow Pkts/s	1.7289e-01
Flow IAT Min	1.2373e-02
Flow Byts/s	0.217946
TotLen Fwd Pkts	0.035030
Fwd Pkt Len Max	1.1597e-01
Bwd Pkt Len Std	0.050883
Flow IAT Std	2.3611e-02
Bwd Pkt Len Max	0.063556

As illustrated in Figure 4, ID3 and Random Forest are the fastest algorithms for classifying overall attacks and normal samples. In contrast, KNN and MLP have the longest running time. The difficulty of the MLP algorithm increases as the number of instances increases. KNN, which is known as the lazy learner due to its complex train time [44], is the slowest of all algorithms used; it took close to 7000 seconds to go through the classification process, whereas Random Forest and ID3 completed the training and testing in less than 120 seconds. Detection time is a very crucial parameter in network security.

The earlier an attack is detected, the better the chances of avoiding massive damage. The main purpose of this study is to reduce the training time of the algorithms and, at the same time, keep the detection accuracy rate high. To achieve that goal, only a few important features are used to train the algorithms. In terms of detection performance, the MLP classifier attained the lowest F1-Score (0.78), which can be attributed to the total number of hidden layers used (13,13,13). Increasing the hidden layers of an MLP also increases the running time of the model, which, in turn, negatively affects the algorithm's efficiency. The chart in Figure 2 presents the Accuracy and Recall of each ML algorithm, and in Figure 3, the F1Score and Precision levels of ML algorithms are illustrated. It is worth noting that only five features are used in detecting specific attack types. The overall attack and normal samples are classified with only 16 important features, acquiring impressive results. These features are mainly statistical, making them appropriate for anomaly-based detection.

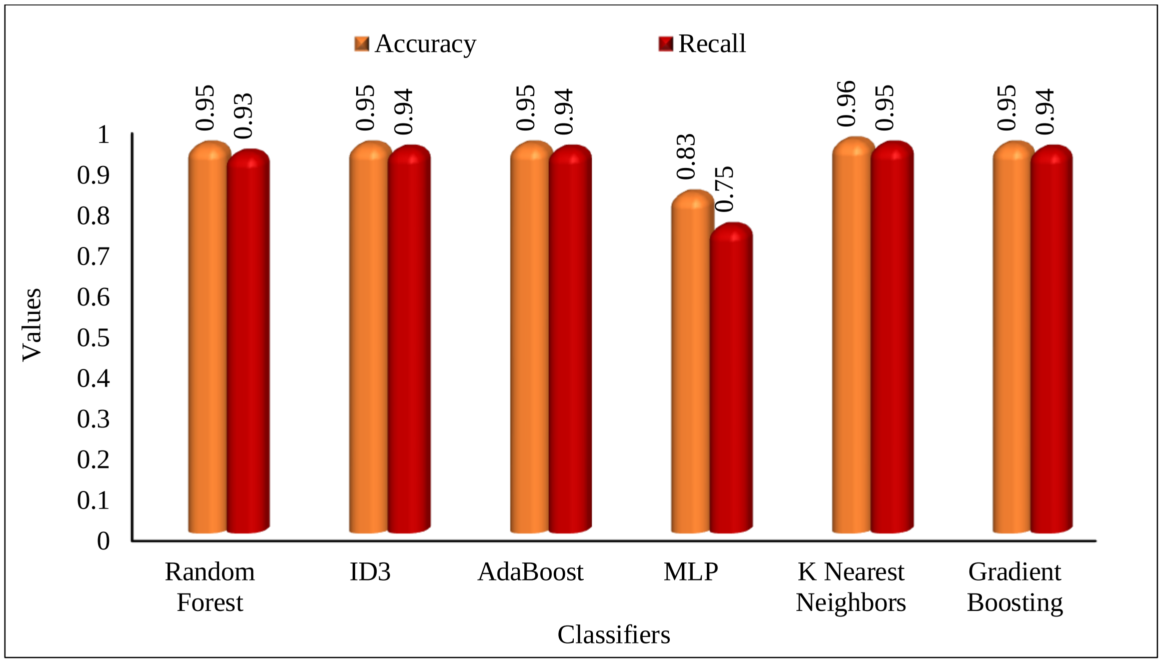

Fig. 2 Accuracy and recall levels achieved by the ML algorithms

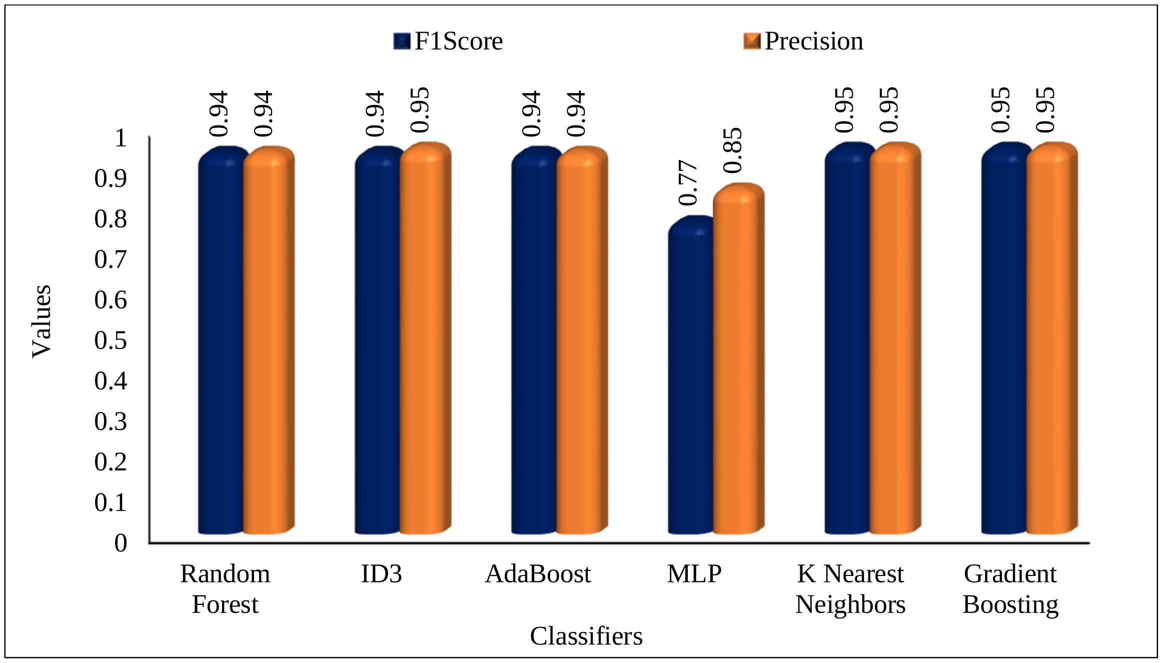

Fig. 3 F1 score and precision levels achieved by the ML algorithms

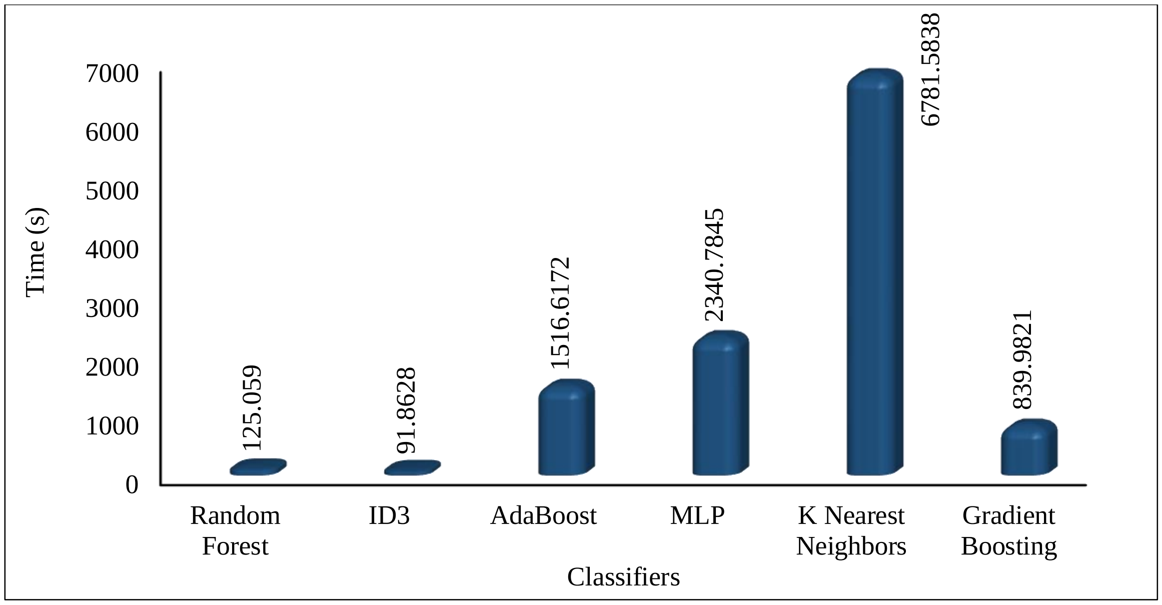

Fig. 4 Running times of ML algorithms

6. Conclusion

This study used six different machine learning algorithms (KNN, XGboost, ID3, MLP, Adaboost, and Random Forest) on the latest IDS dataset (CSE-CIC-IDS2018 dataset) to detect network anomalies by recognizing attack traffic from normal network traffic within a short period of time. The CSE-CIC-IDS2018 dataset constitutes eighty features that explain network traffic flow.

In the first phase of the experimentation process, a dataset sample was cleaned from minor errors. Important feature weights were then calculated with Random Forest Regressor [45]. This was to determine the discriminative features to be used in training machine learning models. The important feature weights for each attack were separately calculated. Next, six machine learning algorithms were trained with the final dataset in two different experiments.

Five out of the six algorithms (XGboost, KNN, ID3, Random Forest, and Adaboost) achieved an F1-Score above 0.93. Hyperparameter tuning can be done to the algorithms in the future to examine how their performance changes in accordance with the parameters. Also, other publicly available IDS datasets could be used to assess model performance.

Funding Statement

Hormuud University funded the research and publication of this article.

References

[1] ITU, Statistics, 2024. [Online]. Available: https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx.

[2] Ali A. Ghorbani, Wei Lu, and Mahbod Tavallaee, *Network Intrusion Detection and Prevention- Concepts and Techniques*, 1st ed., Springer New York, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[3] James F. Kurose, and Keith W. Ross, *Computer Networking: A Top Down Approach*, 7th ed., Pearson, 2017. [Google Scholar] [Publisher Link]

[4] S. Latha, and Sinthu Janita Prakash, "A Survey on Network Attacks and Intrusion Detection Systems," *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, pp. 1-7, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[5] Peter Loshin, Which is Better: Anomaly-Based IDS or Signature-Based IDS?, TechTarget, 2019. [Online]. Available: https://searchsecurity.techtarget.com/tip/IDS-Signature-versus-anomaly-detection

[6] Canadian Institute for Cybersecurity, Applications - CICFlowMeter (Formerly ISCXFlowMeter). [Online]. Available: https://www.unb.ca/cic/research/applications.html#CICFlowMeter

[7] Kaspersky, Brute Force Attack: Definition and Examples, Kaspersky, 2021. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/brute-force-attack

[8] Maryam M. Najafabadi et al., "Machine Learning for Detecting Brute Force Attacks at the Network Level," *2014 IEEE International Conference on Bioinformatics and Bioengineering*, Boca Raton, USA, pp. 379-385, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] Daniel J. Barrett, Richard E. Silverman, and Robert G. Byrnes, *SSH, The Secure Shell - The Definitive Guide*, 1st ed., O'Reilly Media, 2001. [Google Scholar] [Publisher Link]

[10] Riyad Alshammari, and A. Nur Zincir-Heywood, "A Flow Based Approach for SSH Traffic Detection," *2007 IEEE International Conference on Systems, Man and Cybernetics*, Montreal, Canada, pp. 296-301, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[11] Maryam M. Najafabadi et al., "Detection of SSH Brute Force Attacks Using Aggregated Netflow Data," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Miami, USA, pp. 283-288, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[12] GitHub, lanjelot/patator: Patator is a Multi-Purpose Brute-Forcer, with a Modular Design and a Flexible Usage. [Online]. Available: https://github.com/lanjelot/patator

[13] Peter Likarish, Eunjin Jung, and Insoon Jo, "Obfuscated Malicious Javascript Detection Using Classification Techniques," *2009 4th International Conference on Malicious and Unwanted Software (MALWARE)*, Montreal, Canada, pp. 47-54, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[14] Maryam Feily, Alireza Shahrestani, and Sureswaran Ramadass, "A Survey of Botnet and Botnet Detection," *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, Athens, Greece, pp. 268-273, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[15] Arash Habibi Lashkari et al., "A Survey Leading to a New Evaluation Framework for Network Based Botnet Detection," *Proceedings of the 2017 7th International Conference on Communication and Network Security*, pp. 59-66, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[16] Anna Sperotto, and Aiko Pras, "Flow-Based Intrusion Detection," *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, Dublin, Ireland, pp. 958-963, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[17] Canadian Institute for Cybersecurity, CSE-CIC-IDS2018 on AWS, A Collaborative Project between the Communications Security Establishment (CSE) & The Canadian Institute for Cybersecurity (CIC). [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[18] G. Carl et al., "Denial-of-Service Attack-Detection Techniques," *IEEE Internet Computing*, vol. 10, no. 1, pp. 82-89, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[19] GitHub, jseidl/GoldenEye: GoldenEye Layer 7 (KeepAlive+NoCache) DoS Test Tool. [Online]. Available: https://github.com/jseidl/GoldenEye0

[20] Sunny Behal, and Krishan Kumar, "Characterization and Comparison of DDoS Attack Tools and Traffic Generators - A Review," *International Journal of Network Security*, vol. 19, no. 3, pp. 383-393, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[21] Sergey Shekyan, Tag: Slow http Attack, Qualys Community, 2011. [Online]. Available: https://blog.qualys.com/tag/slow-http-attack

[22] Saman Taghavi Zargar, James Joshi, and David Tipper, "A Survey of Defense Mechanisms against Distributed Denial of Service (DDOS) Flooding Attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046-2069, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[23] GitHub, NewEraCracker/LOIC: Low Orbit Ion Cannon - An Open Source Network Stress Tool, Written in C#. Based on Praetox's LOIC Project. [Online]. Available: https://github.com/NewEraCracker/LOIC/

[24] Roxana Papadie, and Ioana Apostol, "Analyzing Websites Protection Mechanisms against DDoS Attacks," *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Targoviste, Romania, pp. 1-6, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[25] Jose Fonseca, Marco Vieira, and Henrique Madeira, "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks," *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, Melbourne, Australia, pp. 365-372, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[26] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, vol. 1, pp. 108-116, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[27] Srilatha Chebrolu, Ajith Abraham, and Johnson P. Thomas, "Feature Deduction and Ensemble Design of Intrusion Detection Systems," *Computers & Security*, vol. 24, no. 4, pp. 295-307, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[28] Weiming Hu, Wei Hu, and Steve Maybank, "AdaBoost-Based Algorithm for Network Intrusion Detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577-583, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[29] Warusia Yassin et al., "Anomaly-Based Intrusion Detection through K-Means Clustering and Naive Bayes Classification," *Proceedings of the 4th International Conference on Computing and Informatics*, pp. 298-303, 2013. [Google Scholar] [Publisher Link]

[30] 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," *ICT Express*, vol. 5, no. 3, pp. 211-214, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[31] Qianru Zhou, and Dimitrios Pezaros, “Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection - An Analysis on CIC-AWS-2018 Dataset,” *arXiv*, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[32] Peng Lin, Kejiang Ye, and Cheng-Zhong Xu, “Dynamic Network Anomaly Detection System by Using Deep Learning Techniques,” *International Conference on Cloud Computing*, pp. 161-176, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[33] Laurens D’hooge et al., “Inter-Dataset Generalization Strength of Supervised Machine Learning Methods for Intrusion Detection,” *Journal of Information Security and Applications*, vol. 54, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[34] Joffrey L. Leevy, and Taghi M. Khoshgoftaar, “A Survey and Analysis of Intrusion Detection Models Based on CSE-CIC-IDS2018 Big Data,” *Journal of Big Data*, vol. 7, pp. 1-19, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[35] Qusyairi Ridho Saeful Fitni, and Kalamullah Ramli, “Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems,” *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, Bali, Indonesia, pp. 118-124, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[36] Gozde Karatas, Onder Demir, and Ozgur Koray Sahingoz, “Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset,” *IEEE Access*, vol. 8, pp. 32150-32162, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[37] XuKui Li et al., “Building Auto-Encoder Intrusion Detection System Based on Random Forest Feature Selection,” *Computers & Security*, vol. 95, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[38] Katherinne Shirley Huancayo Ramos, Marco Antonio Sotelo Monge, and Jorge Maestre Vidal, “Benchmark-Based Reference Model for Evaluating Botnet Detection Tools Driven by Traffic-Flow Analytics,” *Sensors*, vol. 20, no. 16, pp. 1-31, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[39] Abdelraman Alenazi et al., “Holistic Model for HTTP Botnet Detection Based on DNS Traffic Analysis,” *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pp. 1-18, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[40] Richard Zuech, John Hancock, and Taghi M. Khoshgoftaar, “Detecting Web Attacks Using Random Undersampling and Ensemble Learners,” *Journal of Big Data*, vol. 8, pp. 1-20, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[41] Pierre Geurts, Damien Ernst, and Louis Wehenkel, “Extremely Randomized Trees,” *Machine Learning*, vol. 63, pp. 3-42, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[42] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, 2nd ed., Springer New York, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[43] Tianqi Chen, and Carlos Guestrin, “XGBoost: A Scalable Tree Boosting System,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785-794, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[44] Henrik Brink, Joseph W. Richards, and Mark Fetherolf, *Real-World Machine Learning*, Simon and Schuster, 2016. [Google Scholar] [Publisher Link]

[45] Scikit learn, 3.2.4.3.2. sklearn.ensemble.RandomForestRegressor - scikit-learn 0.22.1 Documentation. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[46] William Groves, “Using Domain Knowledge to Systematically Guide Feature Selection,” *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pp. 3215-3216, 2013. [Google Scholar] [Publisher Link]