



Towards the Development of Realistic DoS Dataset for Intelligent Transportation Systems

Rabah Rahal¹ · Abdelaziz Amara Korba¹ · Nacira Ghoualmi-Zine¹

Published online: 6 July 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Vehicular ad-hoc networks (VANETs) present security vulnerabilities, which make them prone to diverse cyberattacks. Denial of Service (DoS) is one of the most prevalent and severe cyberattack that targets VANETs. To tackle this cyberattack and mitigate its effect, intrusion detection systems need to be developed. To this end, a realistic and representative dataset is essential to train and validate the systems. This paper proposes a new dataset, VDoS-LRS, which includes legitimate and simulated vehicular network traffic, along with different types of DoS cyberattack. We also present a realistic testbed environment instead of simulators, taking into consideration different environments (urban, highway and rural). In addition, we explore a wide range of traffic features for detecting and classifying vehicular traffic. We evaluate the reliability of the VDoS-LRS dataset using different machine learning algorithms for forensics purposes. The experimental results showed that it is possible to detect effectively different types of DoS cyberattack within diverse environments.

Keywords ITS · VANET · DoS · Intrusion detection · Network flow · Network forensics

1 Introduction

Intelligent transportation systems (ITS) designate the application of new information and communication technologies in the transport and logistics fields. ITS integrate sensor, control, information and communication technologies to improve road traffic efficiency and safety. Vehicular Ad hoc Networks VANETs are a key part of the ITS framework. Sometimes, VANETs are referred as Intelligent Transportation Networks. Communication between vehicles can be ensured by three modes of communication. The first mode is the Vehicle-to-Vehicle (V2V) communication, which consists on a direct exchange between

✉ Rabah Rahal
rabah.rahal@univ-annaba.org

Abdelaziz Amara Korba
abdelaziz.amara.korba@univ-annaba.org

Nacira Ghoualmi-Zine
ghoualmi@yahoo.fr

¹ Networks and Systems Laboratory (LRS), Badji Mokhtar - Annaba University, Annaba, Algeria

communicating entities [1]. The second mode is the Vehicle-to-Infrastructure (V2I) communication, which requires that each information exchanged between two nodes must have passed through a previously installed infrastructure. The third mode combines both previous modes V2V and V2I [2]. The main objective of this communication is to provide a range of application that can be divided into two categories, Security-related applications, such as cooperative collision warning [V–V], intersection collision warning, approaching emergency vehicle, and work zone warning [R–V] [3]. Moreover, comfort-related applications, such as electronic toll collection, data transfer, parking lot payment, and traffic information [4]. These applications help to increase transportation safety and efficiency and to improve driving conditions for drivers and passengers to make our roads safer [5]. However, these applications raise privacy and security concerns, which significantly threaten the network operation and the user data.

Intrusion detection system based on machine learning (IDS-ML) is one of the known approaches to be effective to protect network against cyberattacks [6]. Based on a classification model, the IDS analyzes a network flow of data packets to check whether there is a suspicious activity in the network, and eventually a cyberattack. Since the effectiveness of the IDS depends on the classification model, it is worthwhile to train the classifier on an realistic and representative network traffic traces dataset that covers a variety of normal and attack samples. However, to the best of our knowledge, such dataset has not been produced for vehicular networks so far. The existing datasets [6–10] contain network traffic traces generated and captured by simulators, which may not be realistic and representative of real environment traffic traces. Therefore, intrusion detection systems trained based on these datasets may not perform effectively in real environment. In this paper, we generate a new vehicular network traffic traces dataset, VDoS-LRS, which contains real network traffic, and diverse types of denial of service cyberattack. We create a realistic testbed environment which considers different types of environment (urban, highway and rural). In addition, we explore a wide range of traffic features for detecting and classifying vehicular traffic. We evaluate the reliability of the VDoS-LRS dataset using different machine learning algorithms for forensics purposes. To be specific, the main contributions of this paper are as follows:

- A new realistic DoS dataset for vehicular networks available upon request, with a detailed description of testbed design and configuration.
- We evaluate the performance evaluation of network forensic methods, based on machine learning algorithms using VDoS-LRS dataset.

The rest of the paper is structured as follows. The literature review of the paper is discussed in Sect. 2. In Sect. 3, we present in details the testbed used to create VDoS-LRS dataset. Section 4 presents the machine learning algorithms used for the classification process. In Sect. 5, we present and discuss experimental results. Section 6 concludes the paper and draws some line for future work.

2 Background and Related Work

This section describes DoS attack scenarios in VANETs networks. Additionally, it outlines the limits of the proposed testbeds and datasets in the literature.

2.1 Denial of Service in Vehicular Networks

In the field of computer security, several descriptions of Denial of Service (DoS) attack can be found. It has been defined by Hasrouny et al. [11] as the attack where “the attacker target the communication medium to cause a channel jam”. DoS is easy to perform, it targets the system’s availability to disable users from accessing to the network. Broadly speaking, DoS attacks can be divided into three types: volume based attack (which saturate the bandwidth of the attacked site like UDP-Flood), protocol based attack (which consumes actual server resources like SYN-Flood) and application layer attack (comprised of seemingly legitimate and innocent requests like Slowloris attack). Quyoom et al. [12] investigated different DoS scenarios in the context of vehicular networks. DoS attack in vehicular networks can target vehicle resources, roadside Units (RSU), and communication channels.

For example, if a normal vehicle attempts to start a TCP connection with an RSU:

- The vehicle ask the establishment of the connection by sending a SYN (synchronize) message.
- The RSU respond to the vehicle by sending back a SYN-ACK message.
- The vehicle send him back an ACK, and the exchange operation can start.

In the case of SYN-Flood (Fig. 1), the malicious vehicle does not respond to the RSU with an ACK, which will push the RSU to wait for the ACK some time to avoid network congestion. Depending on the number of request sent by the attacker, the RSU will be temporarily unavailable to communicate with legitimate vehicles. The attacker can also jams the channel, in a way that vehicles would not be able to access the channel and communicate.

2.2 Existing Network Datasets

There have been many researches in the literature [6–10, 13–20] to protect vehicular networks against cyberattack. Most of them are based on machine learning techniques. Which need datasets for analysing network flows, and modelling normal and malicious network traffic.

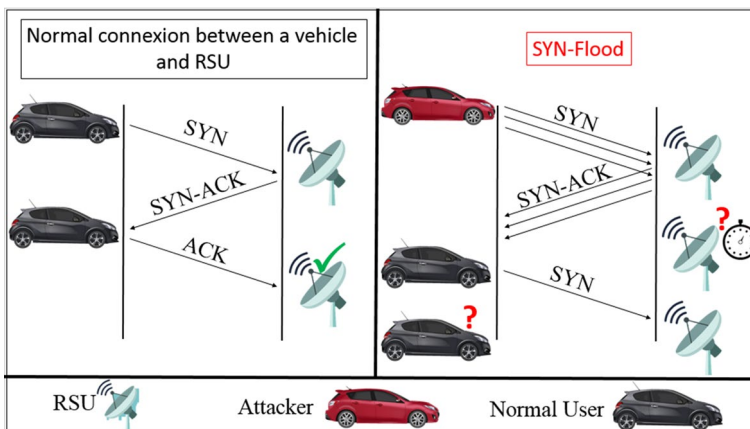


Fig. 1 SYN-Flood in VANET scenario

There have been several datasets containing network traffic, e.g., KDD99 [21], CAIDA datasets [22], NSL-KDD [23], ISCX [24], CICDS2017 [25], UNSW-NB15 [26], which have been used by researchers on intrusion detection and forensic analytics. All the aforementioned datasets lack the inclusion of vehicular traffic data, which make them unusable in the vehicular context.

Although some recent researches [6, 7, 9, 10, 18] have produced synthetic vehicular network datasets for intrusion detection purpose, the development of realistic vehicular network traffic dataset that includes vehicular scenarios is still an unexplored topic. The most recent testbeds and datasets are briefly explained below, with a comparison between them and VDoS-LRS given in Table 1.

Singh et al. [9, 10] proposed a machine learning based approach to detect Wormhole attack in VANETs. For the production of the dataset, they used NS3 simulator [27], which uses the mobility traces generated by the SUMO traffic simulator. They used forty vehicles in the testbed with multi-hop communication using AODV routing protocol. However, in the simulation scenario, it is supposed that all vehicles moved with the same speed during the entire simulation time (15 m/s), which is unrealistic in the vehicular context. Moreover, the duration of the simulation is too short (400 s) to reflect a real vehicular activity.

Alheeti et al. [7] proposed a ML-based IDS to detect grey hole and rushing attacks in a vehicular networks. For the classification, they used SVM and Feed Forward Neural Networks (FFNN). The dataset used to train the IDS was extracted from a trace file generated through simulation. The authors did not provide details about testbed configuration, which makes their simulation difficult to re-product. Grover et al. [8] proposed ML-based approach to classify the node's behaviour, i.e. whether the communicating nodes in the vehicular network is honest or malicious. They tested different classification algorithms: Naive Bayes, IBK, J-48, Random Forest and Ada Boost. However, the proposed IDS has been trained on a network traces dataset collected from simulation using NCTUns-5.0 simulator [28]. In addition, the simulation parameters do not reflect real environment, 6 km area and 2000s simulation duration. Lyamin et al. [18] proposed a data-mining-based approach for real-time detection of radio jamming denial-of-service attacks in the IEEE 802.11p vehicle-to-vehicle (V2V) communications. However, the proposed system was trained statistically for short training sequences of 5 and 100 s, which seems not enough to train an IDS and cover multiple scenarios of radio jamming attack.

Belenko et al. [6] tried to develop a method and a tool to generate close-to-reality datasets for intrusion detection in vehicular networks. The network traces dataset was generated using NS-3 simulator, and includes different types of attacks.

It is worthwhile to see that none of the aforementioned: Belenko et al. [6], Singh et al. [9, 10], Grover et al. [8], and Alheeti et al. [7] researches in the literature used real network traces for the training of the ML-IDS. Knowing that training an IDS with data extracted from a simulator may not be reliable, realistic and representative of vehicular network properties related to real environment. In light of foregoing, this research aims to design a real vehicular network traffic dataset to give a dimension of reliability to our research for detecting DoS attack in the vehicular networks.

Table 1 Synthesis VANET datasets

Dataset	Realistic testbed configuration	Realistic traffic	Take the impact of the environment	Labeled data	VANET traces	Diverse DoS attack scenarios	Full packet capture	New generated features
Belenko et al. [6]	F	F	F	T	T	T	T	T
Singh et al. [9, 10]	F	F	T	T	T	F	F	F
Grover et al. [8]	F	F	F	T	T	F	F	F
Alheeti et al. [7]	F	F	T	T	T	F	T	F
Our work	F	F	F	F	F	F	F	F

Fig. 2 Workflow of DoS detection process

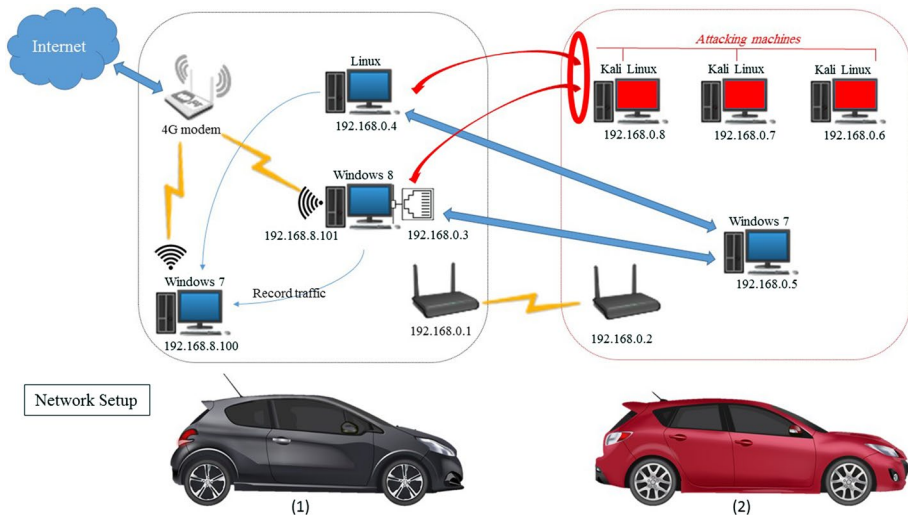
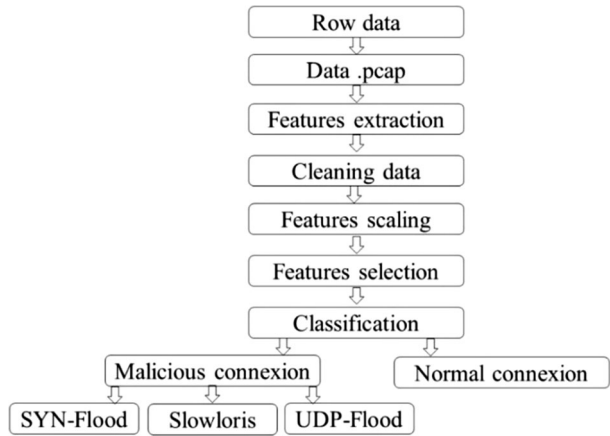


Fig. 3 Testbed environment of the new VDoS-LRS dataset

3 Dataset Generation

In this section, we present in details the followed steps to generate our dataset: testbed configuration, features extraction, data pre-processing and features selection as shown in Fig. 2.

3.1 The Proposed Testbed

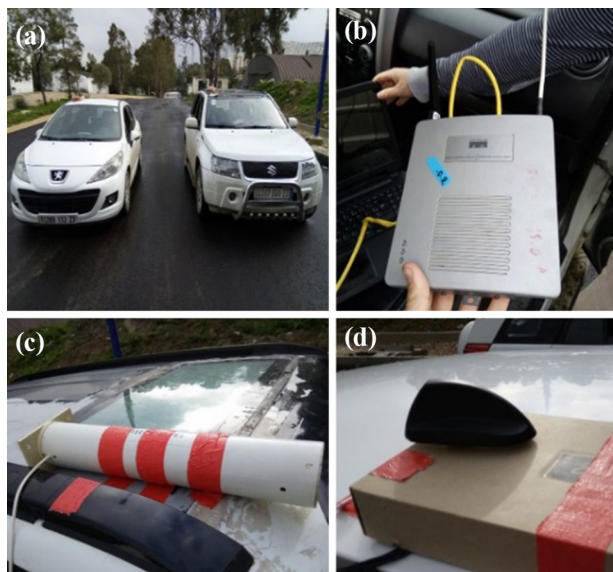
The testbed was made in Annaba, Algeria, we have used: two vehicles, 3 physical machines, 4 virtual machines, 2 access points, 4G modem and 2 Cisco antennas were used

as shown in Fig. 3. Due to resources constraint, we have used two vehicles with multiple virtual machines. The two vehicles (Fig. 4a) were connected in V2V mode using IEEE 802.11G network provided by two AIR-AP1231G-E-K9 access points (Fig. 4b). The access points were connected with two antennas for coverage extension (Sharkee GPSB—Fig. 4d—and Cisco AIR-ANT1949 Antennas—Fig. 4c). This network configuration provides the connection between the two physical machines (Dell computer with an Intel® Pentium® 3558U @ 1.70 GHz 1.70 GHz processor in the first vehicle and Dell computer with Intel® Core i5-4200 U CPU @ 1.60 GHz 2.30 GHz in the second vehicle).

The first machine runs Windows 8 operating system with two network interfaces (Ethernet interface with 192.168.0.3 IP address and Wi-Fi interface with 192.168.8.101 IP address). The second machine runs Windows 7 operating system with IP address 192.168.0.5. Using the Oracle VM virtual Box tool, the first machine executes another Linux system with IP address 192.168.0.4, and the second physical machine (in the second vehicle) executes 3 others virtual machines with following three IP addresses 192.168.0.6, 192.168.0.7, 192.168.0.8, which run the Kali-Linux distribution as the operating system. The three Kali machines represent attackers whose machines execute a different type of DoS attack. The other three machines including two victims.

As shown in Fig. 3, for malicious traffic, the three Kali-Linux machines (in the second vehicle) run a DoS attack taking as a targets the two victim machines (in the first vehicle) with UDP Flood, SYN Flood and Slowloris packets alternately. For benign traffic, we tried to simulate different VANETs services. For example, for the exchange of security information and collaborations between nodes in VANETs, we used packet sender [29], a traffic generator software, which sends UDP, TCP and SSL packets to multiple clients simultaneously. For file sharing between benign nodes, we made a file share using Server Message Block (SMB) protocol that operates as an application layer or presentation-layer network protocol. We also considered common user activities such as access to Google Maps, YouTube, social networks, and real-time applications such as video and audio calls. For data collection, we used another Windows 7 physical machine, which runs a network

Fig. 4 Equipments used in the testbed



traffic analyse called the Wireshark [30]. The experimentation has been conducted within three different environments: urban, rural, and highway. The generated traffic has been stored as pcap files.

We named the produced dataset “VDoS-LRS”, which stands for Vehicular Denial of Service-Networks and Systems Laboratory. The dataset contains normal and DoS malicious traffic. VDoS-LRS took into account three types of DoS attack. SYN Flood exploits vulnerabilities in TCP protocol; it consists on a massive sending of SYN requests to the vehicle. The goal of this attack is to make the vehicle unavailable for legitimate vehicles by exhausting its resources. UDP Flood, which represent the volume-based category, in which the attacker overwhelms random ports on the targeted host with IP packets containing UDP datagrams. The receiving sends back a “Destination Unreachable” packet. As more and more UDP packets are received and answered, the system becomes overwhelmed and unresponsive to other clients. Slowloris, which represent the DoS attacks on the application layers. It can take down another machine with minimal bandwidth and side effects on unrelated services and ports, by keeping many connections to the target opened and hold them open as long as possible (Using uncompleted HTTP request).

In our testbed, we take in consideration the VANETs intrinsic characteristics. Three diverse environments have been considered: urban, highway, and rural. Each environment has its own characteristics. For example, the availability of network coverage in rural environment is not as good as the urban environment. Regarding the vehicle speed, we consider the recommended speeds in each environments (average speed in urban environment 40 km/h, 90 km/h in highway and 30 km/h in rural environment).

3.2 Features Extraction

After collecting the network traffic (as PCAP files) corresponding to the three environments (urban, rural, highway), we extract a set of 79 network features (see Table 2). To extract features we use CICFlowMeter [31], a network traffic flow generator distributed by the Canadian Institut for Cybersecurity (CIC). It generates bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions. Note that TCP flows are usually terminated upon connection teardown (by FIN packet) while UDP flows are terminated by a flow timeout. The flow timeout value can be assigned arbitrarily by the individual scheme e.g., 600 s for both TCP and UDP.

We consider the same five categories of network features as Lashkari et al. [32], which are based on behaviour, bytes, packets, time, and flow. The behaviour-based features (Table 2. [1]) are used to evaluate an object based on its intended actions before it can execute that behaviour. For example, if a connection lasts for a long time, this behaviour may be a behaviour of a DoS attack. The bytes and packets based features (Table 2 [2–13]) are used to count the number of bytes/packets exchanged. The bytes/packets based features allow the detection of large and abnormal traffic increase, which is symptomatic of DoS attacks. In addition, the time taken between the transmission of the packets in a DoS attack is too short (especially TCP Flood and UDP Flood), for this reason, the time-based features can be revealing of DoS attacks (Table 2 [14–17]). Moreover, the flow data features (Table 2 [18–24]) are like packet-based features but with reduced storage space. Instead of on working on packets, we work on a flow of packets. We do not consider source and destination IP addresses and ports, because attackers can change them easily. Furthermore, it may mislead the classification model and prevent it from accurately analyse the rest of the features.

Table 2 List of network features

Features		Description
NB	Name	
1	duration	The duration of the flow in micro-second
2	total_f/b_Packets	The total number of packet in the forward or backward direction
3	total/min/max/mean/std_f/b_Pktl	The total size of packets, the minimum, maximum, mean and the standard deviation size in forward or backward direction
4	f/b_PktsPerSecond	The number of packets per second in the forward or backward direction
5	Avg_Packet_Size	The mean packets size
6	f/b_AvgSegmentSize	The median noticed size in the forward or backward direction
7	f/b_AvgPacketsPerBulk	The mean number of packets bulk rate in the forward or backward direction
8	f/b_AvgBulkRate	The average number of bulk rate in the forward or backward direction
9	act_data_pktl_forward	The number of packets with at least 1 byte of TCP data payload in the forward direction
10	min_seg_size_forward	The smallest segment size noticed in the forward direction
11	Total_f/b_HLen	The total number of bytes used for headers in forward or backward direction
12	f/b_AvgBytesPerBulk	The mean number of bytes bulk rate in the forward or backward direction
13	Init_Win_bytes_forward/backward	The total number of bytes sent in initial window in the forward or backward direction
14	total/min/max/mean/std_f/b_iat	The total, max, min, mean, and standard time between packets for the forward or backward direction
15	f/b_psh/urg_cnt	The number of times the PSH or URG flags were set in packets travelling in the forward or backward direction
16	Min/mean/max/std_active	The min, max, mean, and std time a flow was active before becoming idle
17	Min/mean/max/std_idle	The min, max, mean, and std time a flow was idle before becoming active
18	Flow_Pkts/Bytes_PerSecond	The number of a flow packets or byts per second
19	min/max_flowpktl	The length (min, max) of flow
20	Min/max/mean/std_flowiat	The min, max, mean and std inter-arrival time of packet
21	Flow_fin/syn/rst/psh/ack/urg/cwr/ece	The number of packets with flags
22	downUpRatio	Download and upload ratio

Table 2 (continued)

Features		Description
NB	Name	
23	Sflow_f/b_Packet	The average number of packets in a sub flow for the forward or backward direction
24	Sflow_f/b_Bytes	The average number of packets in a sub flow for the forward or backward direction

3.3 Data Pre-processing and Features Selection

For the data pre-processing step, we did the cleaning and the normalization of data. To check missing values and deal with them, we used one of the python programming language function named Dropna(). Dropna removes a row or a column from a dataframe, which has a NaN or no values in it. Moreover, to deal with the huge differences between magnitude, units, and range in the generated dataset, we used the feature scaling. The feature scaling aims to put all the values in the dataset between 0 and 1, in order to make the features more consistent with each other and to makes the training step less sensitive to this problem. To apply this technique on our dataset, we used from the sklearn library in python the StandardScaler function.

We have used in this research two types of feature selection algorithms. The first one is Forward selection [33], which belongs to the wrappers class. Forward selection is an iterative algorithm that starts with an empty set of feature. In each iteration, it adds the best feature that improves the model until the addition of a new feature does not improve the performance of the model. Forward feature selection algorithm has reduced the number of features from 79 to 10 features as shown in Table 3.

The second features selection algorithm is the Linear Support Vector Classifier LinearSVC [34], which belongs to the embedded features selection algorithms. LinearSVC is an algorithm that gives to each feature a coef\ or feature_importances\ attribute. All the features are considered unimportant at the beginning and it gives them values under a threshold parameter. After that, it uses built-in heuristics for finding the threshold of every feature using a string argument. LinearSVC has reduced the number of features from 79 to 25 as shown in Table 3.

The best results were achieved when using the subset of features giving by the LinearSVC (as shown in the results section).

Table 3 The selected features

Forward selection features set	LinearSVC features set	Common features set
Protocol; Flow Duration Flow IAT Mean; IAT Max Flow Flow IAT Min; Len Bwd Header Len Fwd Header; Syn Flag Cnt Down/Up Ratio; Init Bwd Win Byts	Protocol; Tot Fwd Pkts Tot Bdw Pkts; TotLen Fwd Pkts TotLen Bwd Pkts; Fwd IAT Std Bwd IAT tot; Bwd IAT Max Bwd IAT Min; Fwd Header Len Bwd Header Len; Fwd Pkts/s Bwd Pkts/s; Pkt Len Max Pkt Len Var; Syn Flag Cnt Ack Flag Cnt; Down/Up Ratio Pkt Size Avg; Subflow Fwd Pkts Subflow Fwd Byts; Subflow Bwd Pkts Subflow Bwd Byts Init Fwd Win Byts; Init Bwd win Byts	Protocol; Bwd Header Len Fwd Header Len; Syn Flag Cnt Down/Up Ratio; Init Bwd Win Byts

4 Classification Algorithms

In this section, we briefly present the machine learning algorithms used in our experimentation. We have used the following algorithms for their known efficiency and classification performances: naive Bayes, support vector machine, k-nearest neighbour (KNN), Random Forest and decision trees.

- Naive Bayes Algorithm

An algorithm based on probabilities. A sample probabilistic classifier t , it assumes that given the context of a class. The attributes of an example are independent of each other. Usually also found under the name of “Naive Bayes assumption”. It is based on the theory of Bayes; where the classifier calculates a posteriori probabilities of a class using estimates obtained from a training set if labelled data. In addition, when a new data point is presented for classification the a posteriori probability is calculated for each class, then the example is assigned to the class with the largest posteriori probability [35].

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)}$$

$$P(c|x) = P(x_1|c) * P(x_2|c) * .. * P(x_n|c) * P(c)$$

$P(c|x)$ is the posterior probability of class (target) given predictor (attribute).

$P(c)$ is the prior probability of class.

$P(x|c)$ is the likelihood, which is the probability of predictor given class.

$P(x)$ is the prior probability of predictor.

- Support Vector Machine(SVM)

Algorithms widely used for supervised classification problems, the overall idea and essential key for this classification technique is to find a hyperplane that allows to distinctly classifying data points [36]. Algorithmically, we try to build boundaries between instances of the data set, to transform the original feature space, and separate it by a linear function. This requires a nonlinear separation. To achieve this, SVM algorithms benefits from the concept of distance to find the best margins and the new separation in the new feature space.

- K-Nearest Neighbour

The k-nearest neighbours algorithm (k-NN) is a non-parametric method used for classification and regression [37]. The decision rule in KNN classification is very simple. It works based on minimum distance from the query instance to the training samples to determine the K-nearest neighbours. After gathering the K nearest neighbours, it take simple majority of these K-nearest neighbours to be the prediction of the query instance. Formally:

$$score(D, Ci) = \sum_{Dj \in KNN(d)} Sim(D, Dj) \gamma(Dj, Ci)$$

Above, $KNN(d)$ indicates the set of K-nearest neighbours of the query instance $D * \gamma(Dj, Ci)$ with respect to class Ci , which is:

$$\gamma(Dj, Ci) = \begin{cases} 1, & Dj \in Ci \\ 0, & Dj \notin Ci \end{cases}$$

For test document d , it should be assigned the class that has the highest resulting weighted sum.

- Decision Trees

The concept of this algorithm is to split the data set given a criterion that maximizes the separation of data. The results in a tree like structure [38]. A more simplified idea of the decision tree algorithm is that it breaks down the data set into smaller subsets, at the same time an associated decision tree is incrementally developed, each decision node has two or more branches and the leaf node represents a decision (label of class). There are couple of algorithms to build a decision tree, CART (Classification and Regression Trees) that uses Gini Index (Classification) as metric. Moreover, the one used in this paper, which is ID3 (Iterative Dichotomiser 3) that uses Entropy function and Information gain as metrics. To create a tree, we need to have a root node, which will be the node with the highest information gain in ID3. In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im) purity of an arbitrary collection of examples. Entropy function $H(S)$ can be represented formally as follow:

$$H(s) = \sum_{c \in C} -P(c) * \log_2 P(c)$$

S—The current data set for which entropy is being calculated.

C—Set of classes in S.

P(c)—the proportion of the number of elements in class c to the number of elements in set S.

- Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

In this paper, we used the mentioned algorithms above with the defaults parameters, except with the SVM. For it, we used a nonlinear kernel named the Radial Basis Function (rbf), because it is well in practice and relatively ease for calibrate, as opposed to other kernels.

5 Results and Discussion

In this section, we present and discuss the results of DoS attack detection through two experiments. In the first experiment, we perform binary classification (attack/normal). The second experiment aims to identify the type of the DoS attack (SYN-flood, Slowloris, and UDP-flood). Tables 4 and 5 present samples distribution corresponding respectively to binary and multiclass classification. For both experiences, we have tested several configurations with different sets of network features: (1) the initial features set;

Table 4 Dataset samples distribution for binary classification

Samples	Highway		Rural		Urban	
	Nb	%	Nb	%	Nb	%
Benign	26,480	6.51	9807	3.66	9679	6.30
Malicious	380,002	93.49	257,943	96.34	143,895	93.70
Total	406,482	100	267,750	100	153,574	100

Table 5 Dataset samples distribution for multiclass classification

Samples	Highway		Rural		Urban	
	Nb	%	Nb	%	Nb	%
Benign	26,480	6.51	9807	3.66	9679	6.30
SYN-Flood	157,802	38.82	114,943	42.92	81,188	52.86
Slowloris	1057	0.26	2187	0.81	274	0.17
UDP-Flood	221,143	54.40	140,813	52.59	62,433	40.65
Total	406,482	100	267,750	100	153,574	100

(2) forward selection features set; (3) linearSVC features set; and (4) the common set of features between forward selection and linearSVC. We have used 60% of the dataset for training and 40% for testing.

To evaluate the performances of the proposed DoS detection approach, we used the following performance metrics:

- Accuracy: the ratio of number of correct predictions to the total number of input samples:

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

TP: true positive; TN: True negative; FP: false positive; FN: false negative.

- Precision: represents the proportion of the relevant items among all the items proposed:

$$Precision = \frac{TP}{TP + FP}$$

- Recall: is the fraction of relevant instances that have been retrieved over the total amount of relevant instances:

$$Recall = \frac{TP}{TP + FN}$$

- F1 score: is defined as the harmonic mean between precision and recall. It is used as a statistical measure to rate performance:

$$f1 - score = \frac{2(precision * recall)}{precision + recall}$$

- False positive rate (FPR) is the proportion of all negatives that still yield positive test outcomes:

$$FPR = \frac{FP}{TN + FP}$$

- False negative rate (FNR): is the proportion of positives that yield negative test outcomes with the test:

$$FNR = \frac{FN}{TP + FN}$$

5.1 Experiment 1: Binary Classification

In this experiment, we evaluate the performance of the classifier on classifying a network connection as legitimate or a DoS attack. The classification performances corresponding to the three types of environment (highway, rural, and urban) are presented respectively in Tables 6, 7 and 8. For each environment, we present the classification performances corresponding to the initial features set, and the best set of selected features.

Broadly speaking, the experimental results show that the tree-based algorithms yield the best classification performances within the three types of environment. Decision tree and random forest are known for their high accuracy and stability. On the other hand, the Naive Bayes algorithm performs worse than the other considered algorithms, showing the highest false positive rates. We can see that features selection has slightly affected the performance of decision tree and SVM, by increasing their false positive and negative rates in the case of highway environment. Which is not the case in rural and urban environment, and not the case of the other classification algorithms.

The decision tree classifier yields the best accuracy with the lowest false positive and negative rates in the highway environment.

Whereas, the random forest algorithm outperforms all the other classifiers in the rural and urban environment. The random forest algorithm with the best set of features performs better than with the initial features set. This is mainly due to the fact that in some cases some features might be noisy, redundant and irrelevant, which may mislead the algorithm.

From the obtained results, we can see that the detection precision of malicious traffic was slightly better than the normal traffic. This can be explained by the huge volume of superfluous requests sent by the attacker; the size and the time between packets are completely different from a normal connection. The classification accuracy has been slightly dropped in the highway environment; this may be due to the high speed of vehicles.

From these experiments, we notice that the environment slightly affect the feasibility of denial of service attack in the vehicular context and its detection performances.

5.2 Experiment 2: Multiclass Classification

As in binary classification, the tree-based algorithms show the best classification performances, with similar accuracy scores. However, the decision tree outperforms the random forest in terms of false positive and negatives rates in highway and urban environment. Random forest achieved its best performances with the the LinearSVC selected features set. The Naive Bayes gives the worst results with the lowest accuracy score and the highest false positive and negative rates within the three environment and with both features sets. As we can notice, the environment slightly influences the feasibility of denial of service attack and its detection performances (Tables 9, 10, 11).

Table 6 Binary classification results in highway environment

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Initial set (79)	SVM	Normal	0.99996	0.95825	0.97866	0.0208	0.0208	0.99728
		Malicious	0.99710	0.99999	0.99854			
	Random Forest	Avg. Value	0.99853	0.97912	0.98860			
		Normal	0.99984	0.99977	0.99981	0.0001	0.0001	0.99997
		Malicious	0.99998	0.99998	0.99998			
		Avg. Value	0.99991	0.99988	0.99989			
		Normal	0.90766	0.06540	0.12202	0.4675	0.4675	0.93875
		Malicious	0.93890	0.99953	0.96827			
		Avg. Value	0.92328	0.53247	0.54514			
		Normal	0.99791	0.99353	0.99571	0.0033	0.0033	0.99944
Best set (25)	SVM	Malicious	0.99955	0.99985	0.99970			
		Avg. Value	0.99873	0.99669	0.99771			
	Decision Tree Classification	Normal	0.99988	0.99992	0.99990	0.00004	0.00004	0.99998
		Malicious	0.99999	0.99999	0.99999			
		Avg. Value	0.99994	0.99995	0.99994			
		Normal	1.00	0.95747	0.97827	0.02126	0.02126	0.99722
		Malicious	0.99704	1.00	0.99852			
		Avg. Value	0.99852	0.97873	0.98839			
		Normal	0.99988	0.99981	0.99984	0.00009	0.00009	0.99998
		Malicious	0.99998	0.99999	0.99998			
Naive Bayes	Avg. Value	0.99993	0.99990	0.99991				
	Normal	0.99491	0.94520	0.96942	0.02756	0.02756	0.99611	
	Malicious	0.99619	0.99966	0.99792				
	Avg. Value	0.99555	0.97243	0.98367				
KNN	Normal	0.99856	0.99524	0.99689	0.00242	0.00242	0.99959	

Table 6 (continued)

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
		Malicious	0.99966	0.9999	0.99978			
		Avg. Value	0.99911	0.99757	0.99834			
	Decision Tree Classification	Normal	0.99996	0.99977	0.99986	0.00011	0.00011	0.99998
		Malicious	0.99998	0.99999	0.99999			
		Avg. Value	0.99997	0.99988	0.99992			

Table 7 Binary classification results in rural environment

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Initial set (79)	SVM	Normal	0.99407	0.96002	0.97674	0.02009	0.02009	0.99833
		Malicious	0.99848	0.99978	0.99913			
		Avg. Value	0.99627	0.97990	0.98794			
	Random Forest	Normal	0.99906	0.98711	0.99305	0.00645	0.00645	0.99949
		Malicious	0.99951	0.99996	0.99973			
		Avg. Value	0.99929	0.99354	0.99639			
	Naïve Bayes	Normal	0.96151	0.16349	0.27947	0.41837	0.41837	0.96920
		Malicious	0.96925	0.99975	0.98426			
		Avg. Value	0.96538	0.58162	0.63186			
	KNN	Normal	0.99705	0.96952	0.98310	0.01528	0.01528	0.99878
		Malicious	0.99884	0.99989	0.99936			
		Avg. Value	0.99795	0.98471	0.99123			
Best set (25)	Decision Tree Classification	Normal	0.99989	0.99969	0.99979	0.00015	0.00015	0.99998
		Malicious	0.99998	0.99999	0.99999			
		Avg. Value	0.99994	0.99984	0.99989			
	SVM	Normal	0.99618	0.95758	0.97649	0.02127	0.02127	0.99831
		Malicious	0.99838	0.99986	0.99912			
		Avg. Value	0.99728	0.97872	0.98781			
Random Forest	Normal	1.00	0.99989	0.99994	0.00005	0.00005	0.99999	
	Malicious	0.99999	1.00	0.99999				
	Avg. Value	0.99999	0.99994	0.99997				
Naïve Bayes	Normal	0.77878	0.16692	0.27491	0.41744	0.41744	0.96774	
	Malicious	0.96924	0.99819	0.98350				
	Avg. Value	0.87401	0.58255	0.62921				
KNN	Normal	0.99801	0.97277	0.98523	0.01364	0.01364	0.99893	

Table 7 (continued)

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
		Malicious	0.99896	0.99992	0.99944			
		Avg. Value	0.99848	0.98635	0.99233			
	Decision Tree Classification	Normal	0.99979	0.99979	0.99979	0.00010	0.00010	0.99998
		Malicious	0.99999	0.99999	0.99999			
		Avg. Value	0.99989	0.99989	0.99989			

Table 8 Binary classification results in urban environment

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy	
Initial set (79)	SVM	Normal	0.9977	0.93274	0.96509	0.03363	0.03363	0.99574	
		Malicious	0.99549	0.99998	0.99773				
		Avg. Value	0.99763	0.96636	0.98141				
	Random Forest	Normal	0.99958	0.99979	0.99969	0.00011		0.00011	0.99996
		Malicious	0.99998	0.99997	0.99997				
		Avg. Value	0.99978	0.99988	0.99983				
	Naïve Bayes	Normal	0.97132	0.34642	0.51069	0.32713		0.32713	0.95816
		Malicious	0.95786	0.99931	0.97814				
		Avg. Value	0.96459	0.67286	0.74442				
	KNN	Normal	0.98916	0.98057	0.98485	0.01007		0.01007	0.99809
		Malicious	0.9869	0.9927	0.9898				
		Avg. Value	0.99392	0.98992	0.99191				
Decision Tree Classification	Normal	1.00	0.99989	0.99994	0.00005		0.00005	0.99999	
	Malicious	0.99999	1.00	0.99999					
	Avg. Value	0.99999	0.99994	0.99997					
Best set (25)	SVM	Normal	1.00	0.93377	0.96575	0.03311	0.03311	0.99582	
		Malicious	0.99556	1.00	0.9777				
		Avg. Value	0.99778	0.96688	0.98176				
	Random Forest	Normal	0.99969	1.00	0.99984	0.00001		0.00001	0.99998
		Malicious	1.00	0.99997	0.99998				
		Avg. Value	0.99984	0.99998	0.99991				
Naïve Bayes	Normal	0.98636	0.41853	0.58769	0.29092		0.29092	0.96298	
	Malicious	0.96234	0.99961	0.98062					
	Avg. Value	0.97435	0.70907	0.78416					
KNN	Normal	0.98920	0.98460	0.98690	0.00805	0.00805	0.99835		

Table 8 (continued)

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
		Malicious	0.99896	0.99927	0.99912			
		Avg. Value	0.99408	0.99194	0.99301			
	Decision Tree Classification	Normal	0.99989	0.99989	0.99989	0.00005	0.00005	0.99998
		Malicious	0.99999	0.99999	0.99999			
		Avg. Value	0.99994	0.99994	0.99994			

Table 9 Multiclass classification results in highway environment

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Initial set (79)	SVM	Normal	0.99996	0.95793	0.97849	0.00210	0.24419	0.99482
		SYN Flood	0.98709	1.00	0.99350			
		Slowloris	0.73404	0.06527	0.11989			
		UDP Flood	0.99993	0.99999	0.99996			
		Avg. Value	0.93025	0.75580	0.77296			
	Random forest	Normal	0.99932	0.99932	0.99932	0.00003	0.00608	0.99989
		SYN Flood	0.99991	0.99999	0.99995			
		Slowloris	0.98755	0.97634	0.98192			
		UDP Flood	1.00	1.00	1.00			
		Avg. Value	0.99669	0.99391	0.99529			
Naïve Bayes	Normal	0.97651	0.06593	0.12353	0.05263	0.32196	0.80324	
	SYN Flood	0.92336	0.65006	0.76298				
	Slowloris	0.01453	0.99621	0.02864				
	UDP Flood	0.99996	0.99991	0.99994				
	Avg. Value	0.72859	0.67803	0.47877				
KNN	Normal	0.99844	0.99444	0.99644	0.00019	0.01345	0.99950	
	SYN Flood	0.99924	1.00	0.99961				
	Slowloris	0.98724	0.95175	0.96917				
	UDP Flood	0.99988	0.99999	0.99993				
	Avg. Value	0.99620	0.98654	0.99129				
Decision tree classification	Normal	0.99988	0.99992	0.99990	0.0000003	0.00025	0.99998	
	SYN Flood	1.00	1.00	1.00				
	Slowloris	0.99905	0.99905	0.99905				
	UDP Flood	0.99999	0.99999	0.99999				
	Avg. Value	0.99973	0.99974	0.99973				

Table 9 (continued)

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Best set (25)	SVM	Normal	1.00	0.95751	0.97829	0.00212	0.24430	0.99483
		SYN Flood	0.98704	1.00	0.99348			
		Slowloris	0.73404	0.06527	0.11989			
		UDP Flood	0.99992	1.00	0.99996			
		Avg. Value	0.93025	0.75569	0.77290			
	Random forest	Normal	0.99988	0.99981	0.99984	0.000008	0.00075	0.99998
		SYN Flood	1.00	1.00	1.00			
		Slowloris	1.00	0.99716	0.99857			
		UDP Flood	0.99997	1.00	0.99998			
		Avg. Value	0.99996	0.99924	0.99960			
Naïve Bayes	Normal	0.99736	0.07156	0.13354	0.05230	0.32034	0.80440	
	SYN Flood	0.92475	0.65177	0.76463				
	Slowloris	0.01456	0.99526	0.02871				
	UDP Flood	0.99996	1.00	0.99998				
	Avg. Value	0.73416	0.67965	0.48171				
KNN	Normal	0.99852	0.99539	0.99695	0.00015	0.01321	0.99956	
	SYN Flood	0.99927	1.00	0.99963				
	Slowloris	0.98724	0.95175	0.96917				
	UDP Flood	0.99996	1.00	0.99998				
	Avg. Value	0.99625	0.98678	0.99143				
Decision tree classification	Normal	0.99988	0.99966	0.99977	0.00001	0.00032	0.99997	
	SYN Flood	0.99997	1.00	0.99998				
	Slowloris	1.00	0.99905	0.99952				
	UDP Flood	0.99997	0.99999	0.99998				
	Avg. Value	0.99995	0.99967	0.99981				

Table 10 Multiclass classification results in rural environment

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Initial set (79)	SVM	Normal	0.99534	0.96023	0.97747	0.00346	0.20851	0.99202
		SYN Flood	0.98219	0.99993	0.99098			
		Slowloris	0.99337	0.20576	0.34090			
		UDP Flood	0.99997	1.00	0.99998			
	Random forest	Avg. Value	0.99272	0.79148	0.82733			
		Normal	0.99969	0.99959	0.99964	0.000008	0.00033	0.99997
		SYN Flood	0.99998	0.99999	0.99998			
		Slowloris	0.99908	0.99908	0.99908			
	Naïve Bayes	UDP Flood	1.00	1.00	1.00			
		Avg. Value	0.99969	0.99966	0.99967			
		Normal	0.98500	0.26797	0.42132	0.04413	0.26530	0.83281
		SYN Flood	0.96416	0.67328	0.79288			
	KNN	Slowloris	0.04955	0.99771	0.09442			
		UDP Flood	1.00	0.99981	0.99990			
		Avg. Value	0.74968	0.73469	0.57713			
		Normal	0.99727	0.97144	0.98419	0.00049	0.01608	0.99860
Decision tree classification	SYN Flood	0.99841	0.99988	0.99914				
	Slowloris	0.93401	0.96433	0.94893				
	UDP Flood	0.99988	0.99997	0.99993				
	Avg. Value	0.98239	0.98391	0.98305				
Decision tree classification	Normal	0.99959	0.99989	0.99974	0.000004	0.00037	0.99998	
	SYN Flood	1.00	0.99999	0.99999				
	Slowloris	0.99954	0.99862	0.99908				
	UDP Flood	1.00	1.00	1.00				
Avg. Value	0.99978	0.99962	0.99970					

Table 10 (continued)

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Best set (25)	SVM	Normal	0.99618	0.95758	0.97649	0.00349	0.20916	0.99195
		SYN Flood	0.98196	1.00	0.99090			
		Slowloris	0.99337	0.20576	0.34090			
		UDP Flood	0.99997	1.00	0.99998			
		Avg. Value	0.99287	0.79083	0.82707			
	Random forest	Normal	0.99959	0.99989	0.99974	0.000005	0.00025	0.99998
		SYN Flood	1.00	0.99998	0.99999			
		Slowloris	1.00	0.99908	0.99954			
		UDP Flood	0.99999	1.00	0.99999			
		Avg. Value	0.99989	0.99974	0.99981			
Naïve Bayes	Normal	0.99687	0.19486	0.32599	0.04164	0.28079	0.83629	
	SYN Flood	0.99310	0.68745	0.81248				
	Slowloris	0.04785	0.99451	0.09131				
	UDP Flood	0.99999	1.00	0.99999				
	Avg. Value	0.75945	0.71920	0.57744				
KNN	Normal	0.99790	0.97297	0.98528	0.00045	0.01557	0.99868	
	SYN Flood	0.99846	0.99992	0.99919				
	Slowloris	0.93404	0.96479	0.94916				
	UDP Flood	0.99996	1.00	0.99998				
	Avg. Value	0.98259	0.98442	0.98340				
Decision tree classification	Normal	0.99959	0.99949	0.99954	0.00001	0.00036	0.99996	
	SYN Flood	0.99998	0.99999	0.99998				
	Slowloris	0.99908	0.99908	0.99908				
	UDP Flood	0.99999	0.99999	0.99999				
	Avg. Value	0.99966	0.99963	0.99965				

Table 11 Multiclass classification results in urban environment

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Initial set (79)	SVM	Normal	0.99988	0.93294	0.96525	0.00299	0.21567	0.99434
		SYN Flood	0.98950	0.99998	0.99471			
		Slowloris	1.00	0.20437	0.33939			
		UDP Flood	0.99990	1.00	0.99995			
		Avg. Value	0.99732	0.78432	0.82483			
	Random forest	Normal	0.99865	0.99989	0.99927	0.00002	0.01188	0.99990
		SYN Flood	0.99998	1.00	0.99999			
		Slowloris	1.00	0.95255	0.97570			
		UDP Flood	1.00	1.00	1.00			
		Avg. Value	0.99966	0.98811	0.99374			
Naïve Bayes	Normal	0.99076	0.38805	0.55768	0.06680	0.25981	0.73746	
	SYN Flood	0.98788	0.57641	0.72803				
	Slowloris	0.00682	0.99635	0.01356				
	UDP Flood	0.99996	0.99993	0.99995				
	Avg. Value	0.74636	0.74018	0.57480				
KNN	Normal	0.98907	0.98233	0.98569	0.00077	0.10387	0.99816	
	TCP Flood	0.99809	0.99998	0.99904				
	Slowloris	0.97633	0.60218	0.74492				
	UDP Flood	0.99971	0.99998	0.99984				
	Avg. Value	0.99080	0.89612	0.93237				
Decision tree classification	Normal	0.99989	0.99969	0.99979	0.000008	0.00008	0.99997	
	SYN Flood	0.99998	0.99998	0.99998				
	Slowloris	0.99275	1.00	0.99636				
	UDP Flood	1.00	1.00	1.00				
	Avg. Value	0.99815	0.99991	0.99903				

Table 11 (continued)

Nb. of features	Algorithm	Class	Precision	Recall	F1-score	Avg. FPR	Avg. FNR	Avg. Accuracy
Best set (25)	SVM	Normal	1.00	0.93367	0.96569	0.00296	0.21548	0.99440
		SYN Flood	0.98963	1.00	0.99479			
		Slowloris	1.00	0.20437	0.3939			
		UDP Flood	0.99983	1.00	0.9991			
		Avg. Value	0.99736	0.78451	0.82495			
	Random forest	Normal	0.99969	1.00	0.99984	0.000005	0.00273	0.99998
		SYN Flood	1.00	1.00	1.00			
		Slowloris	1.00	0.98905	0.99449			
		UDP Flood	1.00	1.00	1.00			
		Avg. Value	0.99992	0.99726	0.99858			
Naïve Bayes	Normal	1.00	0.43010	0.60150	0.06530	0.25009	0.74034	
	SYN Flood	0.99676	0.57680	0.73074				
	Slowloris	0.00680	0.99270	0.01350				
	UDP Flood	0.99998	1.00	0.99999				
	Avg. Value	0.75088	0.74990	0.58643				
KNN	Normal	0.98911	0.98605	0.98758	0.00065	0.10293	0.99841	
	SYN Flood	0.99838	1.00	0.99919				
	Slowloris	0.97633	0.60218	0.74492				
	UDP Flood	0.99993	1.00	0.99996				
	Avg. Value	0.99094	0.89706	0.93291				
Decision tree classification	Normal	0.99958	0.99989	0.99974	0.00001	0.00004	0.99996	
	SYN Flood	0.99998	1.00	0.99999				
	Slowloris	1.00	1.00	1.00				
	UDP Flood	1.00	0.99993	0.99996				
	Avg. Value	0.99989	0.99995	0.99992				

Regarding the detection precision of each attack type, we can notice that SYN-Flood and UDP-Flood attacks are relatively easier to detect than the Slowloris attack by all the classifiers. This can be explained by the fact that the Slowloris modus operandi is quite different from the others DoS attacks. Slowloris requires minimal bandwidth and does not flood the server. Periodically, the attacker sends subsequent headers for each request, but never completes it. In contrast to SYN-Flood and UDP-Flood attacks, which requires maximum bandwidth. Those differences make the Slowloris DoS harder to detect compared to SYN-Flood and UDP-Flood.

6 Conclusion

In this paper, we proposed a new dataset, VDoS-LRS, which includes normal vehicular ad hoc network traffic, and various types of denial of service attacks traffic. This dataset was generated and labelled based on a realistic testbed, which takes into consideration three types of environments (urban, rural and highway). To characterize network flows, we extracted from the raw network traffic traces a set of eighty features related to session behaviour, exchanged bytes/packets, and time intervals. Then we carried out feature selection to obtain the best set of features, before the training and testing phase. Afterwards, we evaluate the performances of the initial and selected feature sets with five common machine-learning algorithms. The experimental results showed that the decision tree classifier yields the highest accuracy with the lowest false positive and negative rates within the three environments. The detection system based on this classifier would be able to detect DoS attack in VANET as well as its types with an accuracy of 99%. We observed that the environment slightly affects the feasibility and detection performances of DoS attack. In the highway environment, the classifier needs all the features to give the same performances as in the other environments. In future work, we will consider more denial of service scenarios along with other attacks in VANETs. In addition, we plan to extend our testbed to include more vehicles.

References

1. Rehman, O. M. H., Bourdoucen, H., & Ould-Khaoua, M. (2015). Forward link quality estimation in VANETs for sender-oriented alert messages broadcast. *Journal of Network and Computer Applications*, 58, 23–41.
2. Mihaita, A.-E., Dobre, C., Pop, F., Mavromoustakis, C. X., & Mastorakis, G. (2016). Secure opportunistic vehicle-to-vehicle communication. In C. X. Mavromoustakis, G. Mastorakis, C. Dobre (Eds.), *Studies in big data* (pp. 229–268). Springer.
3. Bektache, D., Tolba, C., & Zine, N. G. (2014). Forecasting approach in VANET based on vehicle kinematics for road safety. *International Journal of Vehicle Safety*, 7(2), 147.
4. Gadkari, M. Y., & Sambre, N. B. (2012). VANET: Routing protocols, security issues and simulation tools. *IOSR Journal of Computer Engineering*, 3(3), 28–38.
5. Malik, A., & Pandey, B. (2018). CIAS: A comprehensive identity authentication scheme for providing security in VANET. *International Journal of Information Security and Privacy (IJISP)*, 12(1), 29–41.
6. Belenko, V., Krundyshev, V., & Kalinin, M. (2018). Synthetic datasets generation for intrusion detection in VANET. In *Proceedings of the 11th international conference on security of information and networks* (p. 9). ACM.
7. Alheeti, K. M. A., Gruebler, A., & McDonald-Maier, K. D. (2015). On the detection of grey hole and rushing attacks in self-driving vehicular networks. In *2015 7th Computer science and electronic engineering conference (CEECE)* (pp. 231–236). IEEE.

8. Grover, J., Prajapati, N. K., Laxmi, V., & Gaur, M. S. (2011). Machine learning approach for multiple misbehavior detection in VANET. In *International conference on advances in computing and communications* (pp. 644–653). Berlin: Springer.
9. Singh, P. K., Gupta, R. R., Nandi, S. K., & Nandi, S. (2019). Machine learning based approach to detect wormhole attack in VANETs. In *Workshops of the international conference on advanced information networking and applications* (pp. 651–661). Cham: Springer.
10. Singh, P. K., Gupta, S., Vashistha, R., Nandi, S. K., & Nandi, S. (2019). Machine learning based approach to detect position falsification attack in VANETs. In *International conference on security and privacy* (pp. 166–178). Singapore: Springer.
11. Hasrouny, H., Samhat, A. E., Bassil, C., & Laouiti, A. (2017). VANet security challenges and solutions: A survey. *Vehicular Communications*, 7, 7–20.
12. Quyoom, A., Ali, R., Gouttam, D. N., & Sharma, H. (2015). A novel mechanism of detection of denial of service attack (DoS) in VANET using Malicious and Irrelevant Packet Detection Algorithm (MIPDA). In *International conference on computing, communication and automation* (pp. 414–419). IEEE.
13. Aloqaily, M., Otoum, S., Al Ridhawi, I., & Jararweh, Y. (2019). An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 90, 101842.
14. Anzer, A., & Elhadeif, M. (2018). Deep learning-based intrusion detection systems for intelligent vehicular ad hoc networks. In J. J. Park, V. Loia, K.-K. R. Choo, & G. Yi (Eds.), *Advanced multimedia and ubiquitous engineering* (pp. 109–116). Singapore: Springer.
15. Gyawali, S., & Qian, Y. (2019). Misbehavior detection using machine learning in vehicular communication networks. In *ICC 2019–2019 IEEE international conference on communications (ICC)* (pp. 1–6). IEEE.
16. Krundyshev, V., Kalinin, M., & Zegzhda, P. (2018). Artificial swarm algorithm for VANET protection against routing attacks. In *2018 IEEE industrial cyber-physical systems (ICPS)* (pp. 795–800). IEEE.
17. Kumar, S., & Mann, K. S. (2018). Detection of multiple malicious nodes using entropy for mitigating the effect of denial of service attack in VANETs. In *2018 4th International conference on computing sciences (ICCS)* (pp. 72–79). IEEE.
18. Lyamin, N., Kleyko, D., Delooz, Q., & Vinel, A. (2019). Real-time jamming DoS detection in safety-critical V2V C-ITS using data mining. *IEEE Communications Letters*, 23(3), 442–445.
19. Schmidt, D. A., Khan, M. S., & Bennett, B. T. (2019). *Spline based intrusion detection in vehicular ad hoc networks (VANET)*. arXiv preprint [arXiv:1903.08018](https://arxiv.org/abs/1903.08018).
20. Tomanđl, A., Fuchs, K. P., & Federrath, H. (2014). REST-Net: A dynamic rule-based IDS for VANETs. In *2014 7th IFIP wireless and mobile networking conference (WMNC)* (pp. 1–8). IEEE.
21. Kdd.ics.uci.edu. (1999). *KDD cup 1999 data*. Retrieved August 15, 2019, from <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
22. Analysis, C. (2019). *Data collection, curation and sharing*. [online] CAIDA. Retrieved August 15, 2019, from <https://www.caida.org/data/>.
23. Unb.ca. (2019). *NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. Retrieved August 15, 2019, from <https://www.unb.ca/cic/datasets/nsl.html>.
24. Iscx.ca. (2019). *Datasets—ISCX*. Retrieved August 15, 2019, from <http://www.iscx.ca/datasets/>.
25. Cicans.ca. (2019). *IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. Retrieved August 15, 2019, from <https://www.unb.ca/cic/datasets/ids-2017.html>.
26. Unsw.adfa.edu.au. (2019). *The UNSW-NB15 data set description*. Retrieved August 15, 2019, from <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>.
27. ns-3. (2019). *ns-3*. Retrieved August 15, 2019, from <https://www.nsnam.org/>.
28. Nsl.cs.nctu.edu.tw. (2019). *EstiNet Network Simulator and Emulator (NCTUns)*. Retrieved August 15, 2019, from <http://nsl.cs.nctu.edu.tw/NSL/nctuns.html>.
29. Nagle, D. (2019). *Packet sender—Free utility to for sending/receiving of network packets. TCP, UDP, SSL*. [online] Packetsender.com. Retrieved August 15, 2019, from <https://packetsender.com/>.
30. Wireshark.org. (2019). *Wireshark · Go Deep*. Retrieved August 15, 2019, from <https://www.wireshark.org/>.
31. Netflowmeter.ca. (2019). *NetFlowMeter*. Retrieved August 15, 2019, from <http://netflowmeter.ca/>.
32. Lashkari, A. H., Kadir, A. F. A., Gonzalez, H., Mbah, K. F., & Ghorbani, A. A. (2017). Towards a network-based framework for android malware detection and characterization. In *2017 15th Annual conference on privacy, security and trust (PST)* (pp. 233–23309). IEEE.
33. Raschka, S. (2019). *Sequential feature selector—mlxtend*. [online] Rasbt.github.io. Retrieved August 15, 2019, from http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/.

34. Scikit-learn.org. (2019). *1.13. Feature selection—scikit-learn 0.21.3 documentation*. Retrieved August 15, 2019, from https://scikit-learn.org/stable/modules/feature_selection.html#l1-based-feature-selection.
35. Mitchell, T. M. (1997). *Machine learning* (1st ed.). New York, NY: McGraw-Hill, Inc.
36. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.
37. Tan, S. (2005). Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4), 667–671.
38. Breiman, L. (2017). *Classification and regression trees*. Abingdon: Routledge.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Rahal Rabah is a Doctoral student in Badji Mokhtar University Annaba Algeria. He is a member of the Laboratory of Computer Networks and Systems. His research interests include VANET, Machine learning, computer security, security metric, cryptography, and security analysis.



Abdelaziz Amara Korba received his Ph.D. degree from Badji Mokhtar Annaba University, Algeria, in 2016. His research activities are focused on the following main topics: network security, intrusion and malwares detection, machine learning, and anomaly detection. Currently, he works as associate professor in the department of computer science at the University of Badji Mokhtar Annaba, and he is member of Laboratoire Réseaux et Systèmes (LRS), Badji Mokhtar Annaba University. He does research in cyberattacks detection and prevention in smart grids AMI, and detection of botnet activities in vehicular and IoT environments.

Nacira Ghoualmi-Zine is a Professor in Computer Sciences and has been a Lecturer in the Department of Computer Science at Badji Mokhtar University, Annaba, Algeria since 1985. She is the Head of the Master and Doctoral option entitled Network and Computer Security, and Head of a Laboratory of Computer Networks and Systems. Her research includes cryptography, computer security, intrusion detection system, wireless networks, distributed multimedia applications, quality of service, security in the protocol, and optimisation in networks.