Computer Networks 186 (2021) 107792

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Detection of reduction-of-quality DDoS attacks using Fuzzy Logic and machine learning algorithms☆

Vinícius de Miranda Rios [a,b,*], Pedro R.M. Inácio [b], Damien Magoni [c], Mário M. Freire [b]

[a] Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, AE 310 Sul, AV. LO 5 S/N, 77021-090, Palmas, Tocantins, Brazil
[b] Instituto de Telecomunicações and Departamento de Informática, Universidade da Beira Interior, Rua Marquês de Ávila e Bolama, P-6201-001 Covilhã, Portugal
[c] University of Bordeaux - LaBRI - CNRS, 351, Cours de la Liberation, 33400 Talence, France

ARTICLE INFO

Keywords:
DDoS attack
Low-rate DDoS attack
Reduction-of-Quality DDoS attack
Fuzzy logic
Machine learning algorithms

ABSTRACT

Distributed Denial of Service (DDoS) attacks are still among the most dangerous attacks on the Internet. With the advance of methods for detecting and mitigating these attacks, crackers have improved their skills in creating new DDoS attack types with the aim of mimicking normal traffic behavior therefore becoming silently powerful. Among these advanced DDoS attack types, the so-called low-rate DoS attacks aim at keeping a low level of network traffic. In this paper, we study one of these techniques, called Reduction of Quality (RoQ) attack. To investigate the detection of this type of attack, we evaluate and compare the use of four machine learning algorithms: Multi-Layer Perceptron (MLP) neural network with backpropagation, K-Nearest Neighbors (K-NN), Support Vector Machine (SVM) and Multinomial Naive Bayes (MNB). We also propose an approach for detecting this kind of attack based on three methods: Fuzzy Logic (FL), MLP and Euclidean Distance (ED). We evaluate and compare the approach based on FL, MLP and ED to the above machine learning algorithms using both emulated and real traffic traces. We show that among the four Machine Learning algorithms, the best classification results are obtained with MLP, which, for emulated traffic, leads to a F1-score of 98.04% for attack traffic and 99.30% for legitimate traffic, while, for real traffic, it leads to a F1-score of 99.87% for attack traffic and 99.95% for legitimate traffic. Regarding the approach using FL, MLP and EC, for classification of emulated traffic, we obtained a F1-score of 98.80% for attack traffic and 99.60% for legitimate traffic, while, for real traffic, we obtained a F1-score of 100% for attack traffic and 100% for legitimate traffic. However, the better performance of the approach based on FL, MLP and ED is obtained at the cost of larger execution time, since MLP required 0.74 ms and 0.87 ms for classification of the emulated and real traffic datasets, respectively, where as the approach using FL, MLP and ED required 11'46" and 46'48" to classify the emulated and real traffic datasets, respectively.

1. Introduction

Internet services and online applications have been the target of numerous types of attacks over the years. Among them, DoS (Denial of Service)/DDoS (Distributed Denial of Service) attacks may stop or degrade the services provided to customers, sometimes causing serious financial costs and reputation damage for many online businesses [1–6]. DoS attacks had the initial goal of consuming resources from a system or application, sending a huge amount of poorly formatted data traffic from one single computer towards the victims [7–10]. A classic example of this type of attack is the Ping of Death [11]. With the rise of effective defense techniques against this type of attack, the attackers have redefined it into DDoS which is capable of blocking the resources of the target machine such as the connection link [12], applications [13,14] and/or hardware resources [15]. DDoS attacks aim

☆ This work is funded by Portuguese FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the project UIDB/50008/2020 and by FCT/COMPETE/FEDER under the project SECURIoTESIGN with reference number POCI-01-0145-FEDER-030657, and funded by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competências em Cloud Computing, co-funded by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), and by the Brazilian CAPES Foundation through the grant contract BEX 9095/13-6 of the doctoral student Vinícius de Miranda Rios.

* Corresponding author at: Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, AE 310 Sul, AV. LO 5 S/N, 77021-090, Palmas, Tocantins, Brazil.
E-mail addresses: vinicius.rios@ifto.edu.br (V.d.M. Rios), inacio@di.ubi.pt (P.R.M. Inácio), magoni@labri.fr (D. Magoni), mario@di.ubi.pt (M.M. Freire).
URLs: http://www.ifto.edu.br (V.d.M. Rios), http://www.ubi.pt (P.R.M. Inácio), http://www.u-bordeaux.com (D. Magoni), http://www.ubi.pt (M.M. Freire).

https://doi.org/10.1016/j.comnet.2020.107792
Received 10 March 2020; Received in revised form 19 August 2020; Accepted 29 December 2020
Available online 1 January 2021

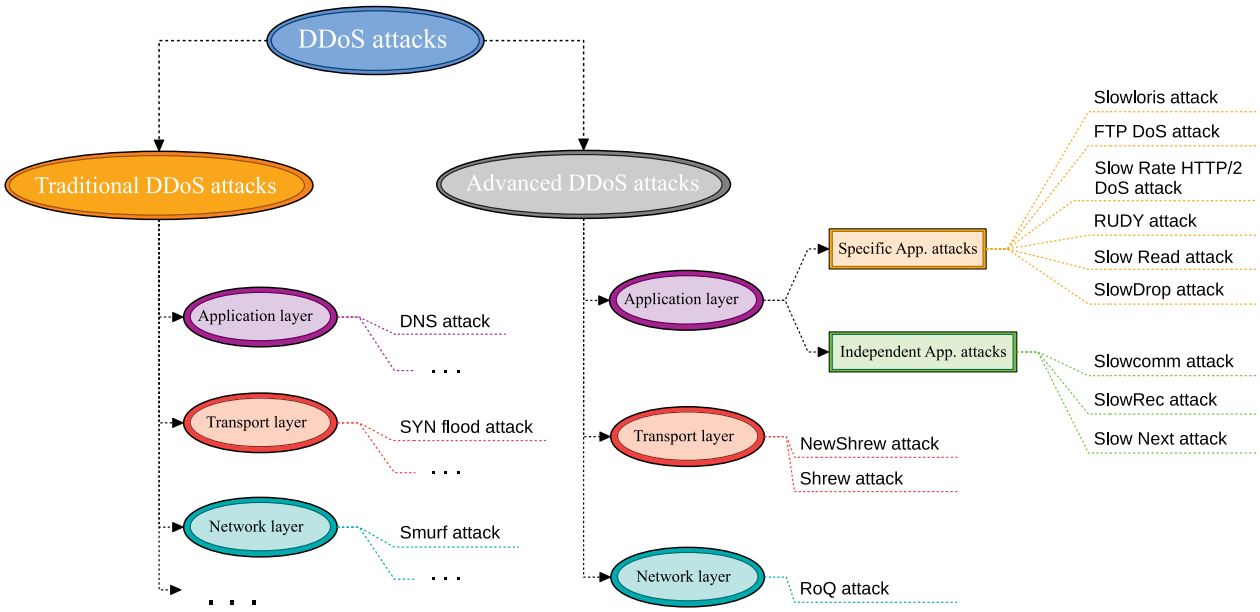

Fig. 1. Classification of DDoS attacks.

at making the old one-to-one attack into a new and more elaborate many-to-one attack, recruiting as many infected computers as possible and creating a botnet (robot network) in order to send a huge amount of data traffic towards the target machine. Some examples of this type of attack include SYN flood attack, Smurf attack, UDP (User Datagram Protocol) flood attack, DNS flood attack (see e.g., [10,16,17]).

In a DDoS attack, the attacker conducts the entire process via handler computers, which through a secure channel such as IRC (Internet Relay Chat) manages the zombie computers that generate the traffic towards the victim in order to compromise its services. Despite DoS/DDoS attacks being a plague to Internet for more than two decades, in 2009 several massive DDoS attacks were carried out in order to disrupt network services of popular websites such as Facebook, Live Journal, Twitter, and Amazon [18]. Since then, the topic of DDoS has been an intense field of research, where new approaches for detection, mitigation, prevention or defense have been proposed, following the evolution of DDoS attack mechanisms (see e.g. surveys [10,16,18–23]) and/or new application domains such as mobile/wireless [24,25], Internet of Things [26], Software-defined networks (SDN) [27,28], cloud [21,22] or fog computing [29,30]. Nevertheless, DDoS attacks have continued to increase and to be successful causing service interruptions that lead to huge financial losses, as was the case of the Dyn attack which took place on October 21, 2016 and was initiated by a DNS operator bringing down major websites including PayPal, Amazon, Airbnb, Visa, The New York Times, Netflix, GitHub, Reddit, Twitter, Spotify, the Guardian, and CNN [31–33]. Other recent high impact attacks include the attack to Dream Host (Hosting Provider) on August 24, 2017 [34], the attack on the UK National Lottery on September 30, 2017 [34], the attack on the Electroneum cryptocurrency startup, just before it launched its mobile mining app on November 2, 2017 [34], the GitHub attack in 2018, which remains the largest DDoS attack of all times, reaching heights when it started at a rate of 1.3Tbps and sent packets at a rate of 126.9 million PPS [33], or the recent attack on the Wikipedia website on September 6, 2019, that paralyzed its site in Europe and some parts of the Middle East [35]. According to a recent report from Neustar Research, major DDoS attacks increased 967% in the first quarter of 2019 compared to the first quarter of 2018 [36], and according to the latest figures of Kaspersky, the number of detected DDoS attacks jumped 18% year-on-year in the second quarter [37].

In order to escape detection, a new kind of DDoS attacks has emerged with the aim of mimicking the legitimate traffic behavior of users of the service. This kind of attack may not cause a service shutdown, but may decrease the quality of the offered service. As a result, DDoS attacks may be classified into two attack types: Traditional Distributed Denial of Service (TDDoS) attacks, which include the classic DDoS attacks, and Advanced Distributed Denial of Service (ADDoS) attacks, which include the new stealthy generation of DDoS attacks. Advanced DDoS attacks may be classified as low-rate DoS/DDoS (also known as LDoS) attacks or as slow DoS/DDoS attacks. Low-rate DoS/DDoS attacks include Reduction of Quality (RoQ) attacks [38–40] at the network layer, Shrew attack [41–43] and the New-Shrew attack [44] at the transport layer, and LoRDAS attack (Low-Rate DoS attacks against Application Servers) [45,46] at the application layer.

Slow DoS/DDoS attacks are confined to the application layer and include the following application-specific attacks: Slowloris (Slow HTTP GET) [47–49], FTP DoS attacks [50], Slow Rate HTTP/2 DoS attacks [14], Slow HTTP POST also known as RUDY (R-U-Dead-Yet) [47], Slow Read attack [51,52], and SlowDrop attack [53]. These attacks have the same format characteristics, but explore different vulnerabilities. Recently, a few DoS attacks have been reported with the characteristic of being independent of the target application layer protocol, which include SlowReq [50,54], Slowcomm [55] and Slow Next [50,56]. An overview of the classification of these DoS/DDoS attacks is proposed in Fig. 1.

This paper focus on the detection of RoQ attacks by using two different approaches based on machine learning algorithms. The first approach consists in the separate use of some machine learning algorithms to detect the RoQ attacks, namely Multi-layer Perceptron (MLP) neural network with backpropagation, K-Nearest Neighbors (K-NN), Support Vector Machine (SVM) and Multinomial Naive Bayes (MNB). These four algorithms have been widely investigated for detecting high-rate DDoS attacks, e, g. MLP used in [57–59], K-NN used in [57,59,60], SVM used in [58–61] and MNB used in [62–64]. The second approach consists in the joint use of a combination of three distinct methods: Fuzzy Loggic (FL), MLP and the Euclidean Distance (ED).

The remaining part of this paper is organized as follows. Section 2 explains how a RoQ attack is performed and Section 3 reviews the related work about detection of RoQ attacks. Section 4 addresses the proposed approaches for the detection of RoQ attacks. Section 5 describes the experimental environments used for the detection of RoQ attacks and Section 6 discusses the obtained results. Section 7 presents our conclusions.

2. RoQ attack

The LDoS (Low-rate Denial of Service) attack aims at reducing the quality of service provided by the target to the point where data transfer rate drops to zero. However the most important characteristic that distinguishes this kind of attack from other DDoS attacks is that, in order to achieve its goal, it does not require a huge amount of data to stop the services running in the target machine. This kind of attack simply sends a quantity of traffic equal to or slightly higher than the bandwidth of the target so that detecting mechanisms do not realize that the ongoing undetected attack causes damage to applications' connections. The RoQ attack seeks to reduce the QoS (Quality of Service) of the target services regardless of the transport protocol used by those services [38,65].

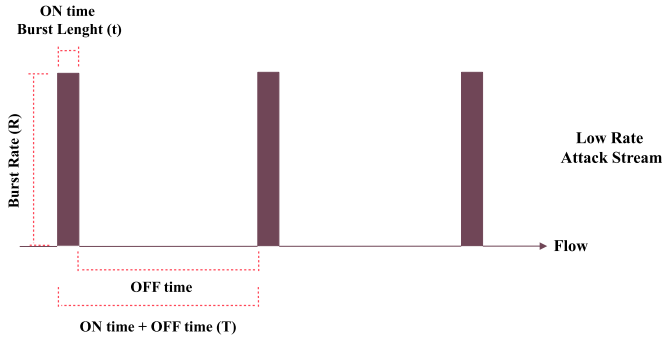

Fig. 2. Schematic representation of the RoQ attack format, where t is the burst length, R is the burst rate and T is the time of the attack period.
Source: Adapted from [66] and [67].

The RoQ attack has a signature that consists of sending a pulse of data packets bigger than or equal to the victim bandwidth in a ON-OFF square wave format so that when it is in ON time the aim is to fill the router queue, sending high traffic rates on longer timescales, forcing the router to drop the legitimate packets passing through it [38,67] as depicted in Fig. 2. This event can cause end system protocols (e.g., TCP) to adapt their sending rates, trying to stabilize the fair sharing network. The OFF time period allows the router to recover from the network traffic shockwave, passing the data packets stuck in the queue through it. Thus allowing applications to continue transmitting data but at a low level, causing a poor QoS of the applications.

3. Related work

This section provides an overview of the use of machine learning algorithms and fuzzy logic for detection of DDoS attacks. It shows the reduced number of works devoted to the detection of advanced DDoS attacks, due to the difficulty in their detection. This section also provides a brief description and a comparative analysis of methods already published for detection of RoQ DDoS attacks.

3.1. Use of machine learning algorithms and Fuzzy logic for detection of DDoS attacks

Soft computing methods (e.g., machine learning, fuzzy logic, evolutionary computation, probabilistic deduction) have been mostly used to detect traditional high-rate DDoS attacks. Khalaf et al. survey in [18] the use of artificial intelligence and statistical approaches for detection of and defense against high-rate DDoS attacks, namely Bayesian networks, fuzzy logic, genetic algorithms, K-NN, neural networks, software agents and support vector machines. Recently, Hosseini and Azizi in [57] have used naive Bayes, random forests, decision trees, multilayer perceptron (MLP), and K-NN for detecting high-rate DDoS. Sreeram and Vuppala in [68] have used machine learning metrics and a bio-inspired bat algorithm for HTTP flood attack detection, Meti et al. in [58] have used machine learning algorithms (Support Vector Machine and Neural Network) for detection of high-rate DDoS attacks in SDNs, and Mohammed et al. have proposed in [69] a machine learning-based collaborative DDoS mitigation mechanism for SDNs. Alrehan and Alhaidari have reviewed in [70] the use of machine learning techniques to detect DDoS attacks on Vehicular Ad hoc NETworks (VANETs), Wani et al. have reported in [61] the use of machine learning algorithms (Support Vector Machine, Naive Bayes, and Random Forest) to detect high-rate DDoS attacks on a cloud environment, Hou et al. have reported in [71] the detection of high-rate DDos attacks through NetFlow analysis using Random Forest, and Aamir and Zaidi in [60] have used K-NN, Support Vector Machine and Random Forest algorithms for high-rate DDoS classification.

Fuzzy logic has also been used for detection of high-rate DDoS attacks. Balarengadurai and Saraswathi in [72] have used fuzzy logic for detection of exhaustion attacks over IEEE 802.15.4 MAC Layer, Rodrìguez et al. have reported in [73] a dynamic DDoS mitigation scheme based on TTL field and fuzzy logic, Mondal et al. in [74] have used fuzzy logic for detection of DDoS in cloud computing environments, and Alsirhani et al. have proposed in [75] a DDoS detection system using a set of classification algorithms (Naive Bayes, Decision Tree (Entropy), Decision Tree (Gini), and Random Forest) controlled by a fuzzy logic system in Apache Spark.

To the best of our knowledge, only a few works have addressed the application of machine learning algorithms for detecting low-rate DDoS attacks. In [13], Singh and De have employed a Multilayer Perceptron with a Genetic Algorithm (MLP-GA) for the detection of Slowloris attacks and in [76] Bhuyan and Elmroth have used a generalized total variation metric to detect Shrew attacks. As far as we know, no work has been reported about the use of fuzzy logic for detection of low-rate DDoS attacks.

3.2. Previous methods for detection of RoQ attacks

This subsection provides a brief description of published methods for detection of RoQ DDoS attacks, as well as their comparative analysis presented in Table 1. This table also puts in evidence how different test-bed components and communication medium characteristics are taken into account by each work and this paper. In this table, the Test-bed Type field represents environments in which attack and detection systems are tested. The Test-bed AQM represents the techniques that align the packets arriving at the router, some techniques being more efficient than others to perform this job. The Detection Mechanism field represents methods that detect and/or respond to an attack. The Protocol field represents the protocol used by the traffic of legitimate clients and attackers. The Attack Software field represents tools that can launch RoQ attacks. Finally, the Performance field represents how the detection mechanisms were effective to detect the attacks. As we can see, this paper has a more complete environment than previous works for testing the whole process of the system.

In order to avoid the loss of quality of voice and video traffic in a MANET (Mobile Ad-hoc Network) by RoQ attack, Ren et al. in [77] have created two defense mechanisms. The first one detects attacks by monitoring the frequency and the retransmission of packets in the MAC layer and the second one responds to the attack by marking the packets with the congestion bit, notifying the emitters. The obtained results show that the higher the amount of attack traffic flows, the greater the delay and the loss of the traffic quality.

To evaluate the RoQ attack against dynamic load balancers, Guirguis et al. in [78] have employed the RoQ attack power metric in order to observe the feedback delay, the resources managed and the average variance in the features in the attack moments, damaging the performance of the entire network. Their results have shown that an attack can cause serious damage against load balancers and they have shown how future defense mechanisms can be created.

Table 1
Comparison among related works and this article.

Works	Testbed environment	AQM testbed	Network type	Mechanisms	Traffic protocol	Attack software	Performance
Ren et al. (2007) [77]	Simulated	Absent	Wireless	To monitor the threshold of three MAC layer signals	TCP and UDP	Absent	Effective
Guirguis et al. (2007) [78]	Simulated and real	Absent	Ethernet	Absent	Absent	Absent	Absent
Guirguis et al. (2007) [79]	Real	Absent	Ethernet	Dynamically adapt β value in admission controller's equation	Absent	Absent	Effective
Chen and Hwang (2007) [80]	Simulated	Droptail	Ethernet	Flow-level spectral analysis with sequential hypothesis testing	TCP and UDP	Absent	Effectively rescue 99% legitimate TCP flows
Shevtekar and Ansari (2008) [67]	Simulated	RED-PD	Ethernet	Router-based approach	TCP and UDP	Absent	Absent
Chen et al. (2008) [81]	Simulated and real	Absent	Wireless	Absent	Absent	Absent	Absent
Arunmozhi and Venkataramani (2010) [82]	Simulated	Absent	Wireless	Flow Monitoring	Absent	Absent	Achieves higher throughput and packet delivery ratio
Gulati and Dhaliwal (2013) [83]	Simulated	Absent	Wireless	Flow Monitoring Table (FMT)	Absent	Absent	Reduce packet loss and improves throughput
Wen et al. (2014) [39]	Simulated and real	Absent	Ethernet	Wavelet multiresolution analysis method with autocorrelation analysis	TCP, ICMP and UDP	absent	High accuracy and high efficiency
Gang et al (2017) [40]	Simulated	Absent	Wireless	Hamilton-path-based scheme	Absent	Absent	Reduce evidently packet loss
Hongsong et al. (2019) [65]	Simulated	Absent	Wireless	Hilbert–Huang Transform (HHT) with Ensemble empirical mode decomposition (EEMD) and Correlated coefficient method	TCP	AOMDV with RREQ	Highly efficient
This article (2020)	Emulated and real	Droptail	Ethernet	Machine learning algorithms, Fuzzy logic and Euclidean distance	TCP and UDP	M-RoQ	See Section 6

Guirguis et al. in [79] have implemented the admission ratio's β parameter value to be dynamically adaptive in the equation:

$$\alpha_i^n = \frac{1}{N} + \beta \sum_{j=1}^{N} (q_{i-1}^j - q_{i-1}^n). \quad (1)$$

In this way, the load balancers can react faster to the changes of the traffic, keeping the services available. The same happens to the admission controllers' K parameter. The results have shown an increased efficiency against the attack. To avoid the servers to be overwhelmed by RoQ attacks, Chen and Hwang in [80] have created a novel defense mechanism by combining the flow-level spectral analysis (which can segregate normal TCP flows from malicious flows) with sequential hypothesis testing. Their results have shown that this new detection system can effectively rescue 99% of the legitimate TCP flows under RoQ attacks.

Shevtekar and Ansari in [67] have addressed the detection of the RoQ attack in two phases. The first detection phase is initialized by the sudden increase of the packets in the queue. Subsequently, the time difference between consecutive arrivals of the packets of each stream is checked; if it is too short, a packet load count is made and if this value is greater than the threshold, the attack is detected. After detection, packets marked as attack packets are discarded. Their results have shown an effective countermeasure approach for RoQ attack and Low-rate DoS attack.

A novel RoQ attack model was proposed by Chen et al. in [81] with the aim of jamming the MAC layer at a special moment, which can repeatedly block TCP ACK transition and degrade wireless TCP throughput. Their results have shown that jamming at the MAC layer can be launched in an actual wireless network, becoming a potential threat to wireless networks around the world.

To stop RoQ attacks on MANETs, Arunmozhi and Venkataramani in [82] have proposed a flow monitoring (FMON) scheme that employs a MAC layer-based detection scheme based on the frequency and retransmission of RTS/CTS and data, as well as a response based on ECN marking. The performance of FMON has been compared to SWAN and SPA-ARA protocols. Their results have shown that FMON outperforms the other two schemes.

To detect RoQ attacks, Gulati and Dhaliwal in [83] have proposed the election of a computer to be the attack monitor. After that, the traffic is monitored and if there is a sudden increase of traffic in a short period of time above a certain threshold, all the nodes connected to that flow will be added to a list of suspects. In the Attack Monitor, the node will be added to a list called checking table if it appears countless times. This table is then sent to all nodes in the network that have verified that their traffic is above the threshold during a certain period of time. If it is positive, the node is added to the attacker table, otherwise it is removed from the checking table. All nodes in the attacker table will have their traffic blocked. Their results have shown that is possible to reduce packet loss and improve throughput.

Wen et al. in [39] have combined anomaly detection with misuse detection to detect potential anomalies through the use of wavelet multi-resolution analysis and auto-correlation analysis. The obtained experimental results show that RoQ attacks were detected accurately with both low false positive and low false negative rates.

Gang et al. in [40] have presented performance tests among MIPv4 (Mobile Internet Protocol version 4), MIPv6 (Mobile Internet Protocol version 6) and FMIPv6 (Fast Mobile Internet Protocol version 6) against RoQ attacks with and without solution. The solution was to use a Hamilton-path-based scheme to decrease the packet loss of the traffic nodes. In the scenario without the proposed solution, the mobile protocol suffers more damage than in the scenario with the proposed solution. Their results have shown that the Hamilton-path-based scheme can decrease packet loss and delay in overlay networks.

Hongsong et al. in [65] have created a novel detection method based on the union of the Hilbert–Huang Transform (HHT) with Ensemble Empirical Mode Decomposition (EEMD) and the Correlated Coefficient Method. This mix of methods aims at eliminating possible problems related to the signal generated by the attack traffic such as mode mixing and false components. To generate the RoQ attack traffic, the Ad hoc On-demand Multi-path Distance Vector (AOMDV) was used with flood Route REQuest (RREQ) messages. Their results have shown a highly efficient detection of the attacks.

4. Proposed methods for detecting RoQ attacks

This section identifies the set of three features to be jointly used by each classifier and describes two different approaches for detection of RoQ DDoS attacks. The first approach, described in Section 4.2, consists on the separate use of four machine learning algorithms. The second approach consists on the use of a combination of three distinct methods: FL, MLP and ED. This second approach is described in Section 4.3.

4.1. Classification features

Given a set of candidate features, the problem of feature selection consists on the selection of a feature or a subset of features that performs the best under some classification algorithms. This process can reduce not only the cost of recognition by reducing the number of features, but also provide a better classification accuracy due to finite dataset size effects [84]. Although the feature selection process is useful when we have a large number of features [85–87], it also may impose severe restrictions for real-time operation. Therefore, in this paper, we followed an heuristic approach of manually selecting features that lead to good classification performance without compromising real-time operation, as followed in other approaches, such as in [13,88] where the authors used the number of packets for detecting DDoS attacks, [43,89–92] where the authors used entropy for DDoS attack detection, [93] where the authors used average inter-arrival time in a defense mechanism against TCP SYN flood DDoS attack, or [94] where the authors used a combination of entropy, number of packets and average inter-arrival time for detection of encrypted peer-to-peer traffic.

We consider three features for classification purposes: number of packets, entropy and average of inter-arrival time. Since, in our preliminary research study, each of these three features used individually led to poor classification results, except entropy, which for some cases led to good classification results as we can see in Section 6.3, we explore in this paper the joint use of these three features for RoQ attack detection.

The number of packets in a given flow is chosen as a feature because the amount of data flows may grow substantially even in a low-rate DoS attack.

The entropy measures the degree of uncertainty information associated with a random variable [95]. The more uncertain the result of a random experiment is, the more information is obtained by observing its occurrence. In this work, we evaluate the entropy of the quintet consisting of source IP address, source port, destination IP address, destination Port and transport protocol (TCP and UDP) for the network traffic. Therefore, the higher the randomness of the source IP address and the source port fields the greater the entropy and conversely, the more constant the source IP address and the source port fields, the smaller the entropy. The entropy H is given by [95]:

$$H = -\sum_{i=1}^{n} p_i log p_i. \quad (2)$$

where p_i is the probability of each quintet element occurring in the time window under evaluation of the traffic trace and n is the total number of packets in the time window of the trace. If the flow does not have source neither destination port fields in the quintet, it is only composed of source and destination IP addresses and protocol in a given time window of the trace.

The third feature, the average of inter-arrival time of packets is selected because in flood attacks the packets do not wait in a queue to be sent by the network card. Instead, they are sent as quickly as possible, which result in packets with smaller RTT (Round Trip Time) than legitimate packets.

These three features are computed as follows. For a certain traffic trace, we apply a sliding time window with length ΔT over the time in the Time Stamp field of the trace in the tshark tool. In this work, ΔT assumes a default value of 1 s. At the beginning, for a sliding time window with length of 1 s, we evaluate these three features for all packets in the first second of the traffic trace. Therefore, we count the number of packets within the first second of the trace, we evaluate the entropy as described above for the packets within the first second of the dataset, and we evaluate the average of inter-arrival times of packets within the first second in the trace. Then, we slide the time window 1 s forward, and evaluate the number of packets, entropy and average of inter-arrival times for the packets within the second (sliding time window with length of 1 s) of the trace. This process is repeated by sliding the time window with length ΔT until it reaches the end of the dataset. The last time window may have a length smaller than ΔT because the length of the traces usually is not a multiple of ΔT.

Using the above procedure, for a given traffic trace, we obtain, for all time windows with length ΔT, the values of the three features for the packets within each time window of the traffic trace. The set of these three features for all time windows of the trace is named traffic dataset. For performance assessment purposes of the classifiers, besides the values of the three features for each time window, we also have the information about the traffic (legitimate or attack) per time window to serve as ground-truth, having this information been obtained during the construction of the traffic trace in a controlled environment. The classification approaches described along next subsections uses the traffic dataset to classify the traffic.

4.2. Machine learning algorithms and their settings

The following machine learning algorithms are considered in this work to investigate the detection of RoQ attacks: Multi-layer Perceptron (MLP) neural network with backpropagation, K-Nearest Neighbors (K-NN), Support Vector Machine (SVM) and Multinomial Naive Bayes (MNB).

Table 2
Machine learning algorithms and their settings.

Algorithm	Settings
K-NN	n_neighbors = 5, weights = uniform, algorithm = auto, leaf_size = 30, p = 2, metric = minkowski, metric_params = None, n_jobs = None
MLP	activation = relu, alpha = 1e−05, batch_size = auto, beta_1 = 0.9, beta_2 = 0.999, early_stopping = False, epsilon = 1e−08, hidden_layer_sizes = (5, 2), learning_rate = constant, learning_rate_init = 0.001, max_iter = 200, momentum = 0.9, nesterovs_momentum = True, power_t = 0.5, random_state = 1, shuffle = True, solver = 'lbfgs', tol = 0.0001, validation_fraction = 0.1, verbose = False, warm_start = False
SVM	kernel = rbf, degree = 3, gamma = auto, coef0 = 0.0, tol = 0.001, C = 1.0, epsilon = 0.1, shrinking = True, cache_size = 200, verbose = False, max_iter = −1
MNB	alpha = 1.0, fit_prior = True, class_prior = None

These four algorithms are separately used to classify Internet traffic as legitimate traffic or attack traffic, jointly with three features evaluated for each traffic dataset. The implementation of these four algorithms at scikit-learn [96] has been used with their default configurations. The four algorithms and their settings used in this work are summarized in Table 2. The training dataset, with the values of the three features per time window for the training trace described in Section 5, has been used for the training phase of these algorithms.

4.3. Approach based on fuzzy logic, MLP and Euclidean distance

This approach is based on a combination of three methods: FL, MLP and ED. Fuzzy logic was introduced by Zadeh in 1965 [97]. It is a mathematical theory applied to vague concepts that admit intermediate logical values between false and true (0 or 1) in regard to elements belonging to a certain set with a certain pertinence degree, giving a mathematical treatment to subjective linguistic terms.

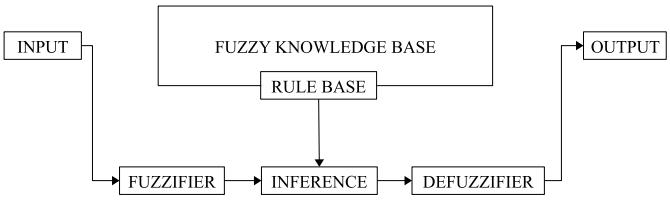

Fig. 3. Schematic representation of fuzzy expert system.
Source: Adapted from [74].

In order to explain how a fuzzy expert system works, we divide it into five main parts, as illustrated in Fig. 3. The first part is the input, which receives the numerical data in which the system relies on to make decisions. The second part is the fuzzification, which transforms the input data into fuzzy information. The third part is the fuzzy inference module, which includes the knowledge base and the logical decision maker. The fourth part is the defuzzification which transforms the fuzzy inference system output into numerical information presented at the output of the system.

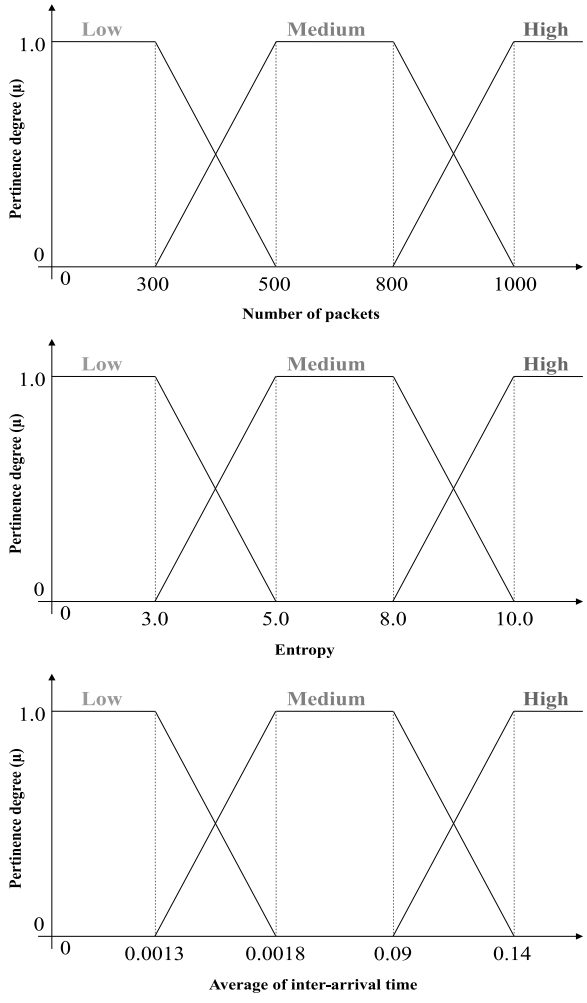

Fig. 4. Pertinence function for the features number of packets, entropy and average of inter-arrival time (ms).

The fuzzification is the process of normalizing the input data through pertinence functions, turning quantitative values into qualitative values such as “very low”, “low”, “medium”, “high”, “very high” in the universe of discourse of a certain input variable. Therefore, the normalized datum (pertinence degree) can belong to more than one linguistic term, i.e., sometimes the same input datum can be classified, for example, as “low” and “medium” at the same time.

The inference module constructs rules that are presented in the form “if … then”, describing the action to be taken in response to several fuzzy normalized data outputs. It has the objective of creating a knowledge base of rules to help in the decision making, in order to obtain an accurate final result. Finally, the defuzzification process of the output data of the inference module is performed by one of the available defuzzification methods to be chosen, such as first of maxima, center of area, middle of maxima and others.

In this work, the fuzzy system is configured using three linguistic terms, namely “low”, “medium” and “high”, for each of the features number of packets, entropy and average inter-arrival time. The pertinence function chosen to normalize the data values is trapezoidal for all features as depicted in Fig. 4. The range of values selected for the linguistic terms in the universe of discourse for each variable is based on the values of the features (number of packets, entropy and average inter-arrival time) evaluated from the datasets. Therefore, the lowest values of the features are for “low” term, the highest values are for “high” term and the average of these values are for the “medium” term.

The trapezoidal function to normalize data is given by [74]:

$$\mu(x) = \begin{cases} \frac{x-a}{b-a}, \; if \; a \leqslant x < b, \\ 1, \; if \; b \leqslant x < c, \\ \frac{d-x}{d-c}, \; if \; c \leqslant x < d, \\ 0, otherwise. \end{cases} \tag{3}$$

where $\mu(x)$ is the normalized data, x is the data extracted from the datasets and a, b, c and d are the values referring to the data on x axis belonging to highest and lowest pertinence degree. After fuzzification, the values of the pertinence degree generated in each input variable are passed to the set of fuzzy rules. This is formed by 27 rules, which must cover all possible situations of the behavior of the fuzzy system, as we can see in Fig. 5. In all of the rules, the Mamdani inference model [98] is applied, in which the logical operator “AND” is used over the antecedents of each rule, being the lowest value chosen as a consequent among the values of the pertinence degree of the triggered rule.

RULE	IF									THEN
	NUMBER OF PACKETS			ENTROPY			AVERAGE OF INTER-ARRIVAL TIME			CLASSIFICATION
	LOW	MEDIUM	HIGH	LOW	MEDIUM	HIGH	LOW	MEDIUM	HIGH	
1			X			X	X			ATTACK
2			X		X		X			ATTACK
3			X	X			X			LEGITIMATE
4			X			X		X		ATTACK
5			X		X			X		ATTACK
6			X	X				X		LEGITIMATE
7			X			X			X	ATTACK
8			X		X				X	ATTACK
9			X	X					X	LEGITIMATE
10		X				X	X			ATTACK
11		X			X		X			LEGITIMATE
12		X		X			X			LEGITIMATE
13		X				X		X		ATTACK
14		X			X			X		ATTACK
15		X		X				X		LEGITIMATE
16		X				X			X	ATTACK
17		X			X				X	LEGITIMATE
18		X		X					X	LEGITIMATE
19	X					X	X			LEGITIMATE
20	X				X		X			LEGITIMATE
21	X			X			X			LEGITIMATE
22	X					X		X		ATTACK
23	X				X			X		LEGITIMATE
24	X			X				X		LEGITIMATE
25	X					X			X	ATTACK
26	X				X				X	LEGITIMATE
27	X			X					X	LEGITIMATE

Fig. 5. Set of base rules of the inference module.

At the end, the defuzzification module starts after all the rules were triggered by the inference module. In order to transform the values of the pertinence degree, selected as consequent by the inference module, into an accurate output of numerical values, it is necessary to defuzzify them. For this, the middle of maxima method is used, because it is one of the most used in fuzzy expert systems. In this way, based on the

generated results, the system can classify the type of traffic flow. The middle of maxima X^* is given by [74]:

$$X^* = \frac{\sum (maximum\ value * pertinence\ degree)}{\sum pertinence\ degree}. \quad (4)$$

This method aims to identify the maximum value to which belongs the output value of each rule of the inference module, that is, the maximum value of the central point of each linguistic variable of the defuzzification model. After performing the defuzzification, the corresponding output is classified according to the linguistic terms "legitimate" or "attack" in the universe of discourse for the traffic type (legitimate or attack). If the output is smaller or equal than 4.0, the traffic is classified as "legitimate", otherwise, if the output is larger or equal than 6.0, it is classified as "attack". Fig. 6 illustrates this concept.

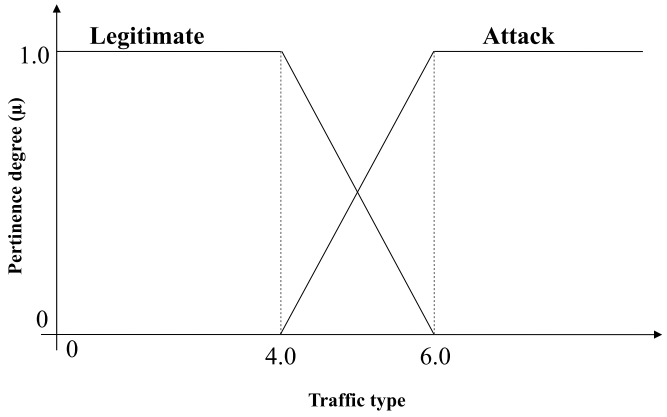

Fig. 6. Defuzzified values of traffic type as legitimate or attack.

After the classification of the traffic in the trace using the FL approach, the MLP algorithm is used, as described in the previous subsection, for a new classification of the traffic in the same trace. Among the four machine learning algorithms, the MLP algorithm has been chosen due to its better performance results presented in Table 9 on Section 6.3.

Both results of the traffic trace classification, for each time window with length ΔT, obtained with FL and MLP approaches are compared. If they are equal, this is the final result of the classification. If they are different, an additional step is performed, which consists in using the ED to obtain the minimum distance between each values of the number of packets and entropy, that lead to different classifications by the two approaches, and all values of the number of packets and entropy in the training dataset. Therefore, we obtain a classification using the ED for the number of packets and another classification for the entropy, since the values of the number of packets and entropy in the training dataset correspond to known traffic previously identified as legitimate or attack. The ED between the value pk (k = 0, …, s, s being an integer smaller than m) of a feature (number of packets or entropy) that leads to different classifications by FL and MLP in the dataset under evaluation and each of the values of the feature pi (i = 1, …, m, where m is the number of values of the feature in the training dataset) is given by:

$$D(p_k, q_i) = \sqrt{(p_k - q_i)^2}. \quad (5)$$

Finally, the classification of the number of packets and the classification of the entropy using the ED is compared to the classifications obtained with the fuzzy logic and neural network approaches. Since we have now four classifications, if the number of classifications of attack type is greater than the classifications as legitimate type, then the final result is an attack and vice versa.

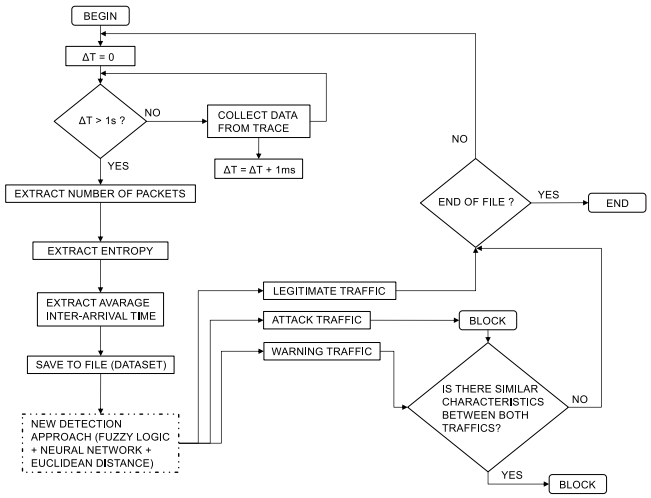

Fig. 7. Flowchart of the classifier for detection of RoQ attacks.

If the two classifications obtained with ED for the number of packets and the entropy are different, then, in the set of four classifications, we have two classifications as attack and two classifications as legitimate, leading to a final result that is considered as a warning. After that, the traffic packets leading to warning alerts are analyzed and compared with the blocked attack traffic packets. If both sets of traffic packets have similar characteristics in terms of the high values of entropy (entropy higher than 9.0), the warning traffic will be blocked. Otherwise, traffic packets leading to warning alerts are forwarded to the destination. The Fig. 7 illustrates the whole classification process for detecting RoQ attacks using the approach based on FL, MLP and ED of two features (number of packets and entropy in a given time window of the trace).

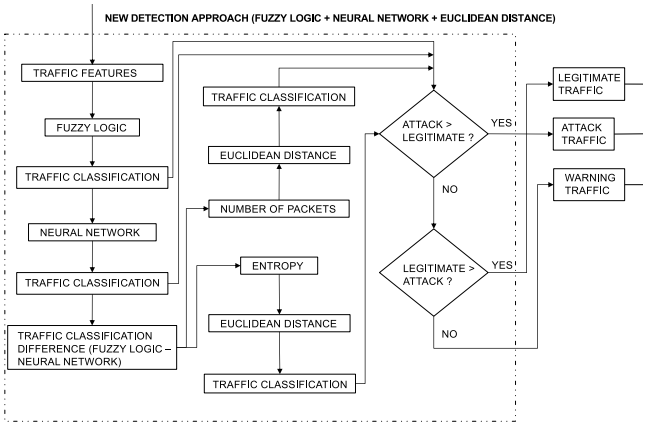

Fig. 8. Flowchart of the classification module using FL, MLP and ED approaches.

The additional step required when the classifications obtained with FL and MLP are different is performed by the module Traffic Classification Module detailed in Fig. 8.

5. Test environment

This section addresses traffic traces and datasets used for classification as well as the emulated and real scenarios used to obtain the traffic traces.

5.1. Traffic traces and datasets

Four Internet traffic traces are considered in this work. Two traces are used for evaluation purposes and were obtained in the emulated and real environments, as described in the next subsections. The trace obtained in the emulated environment has a size of 194.8 MB and the

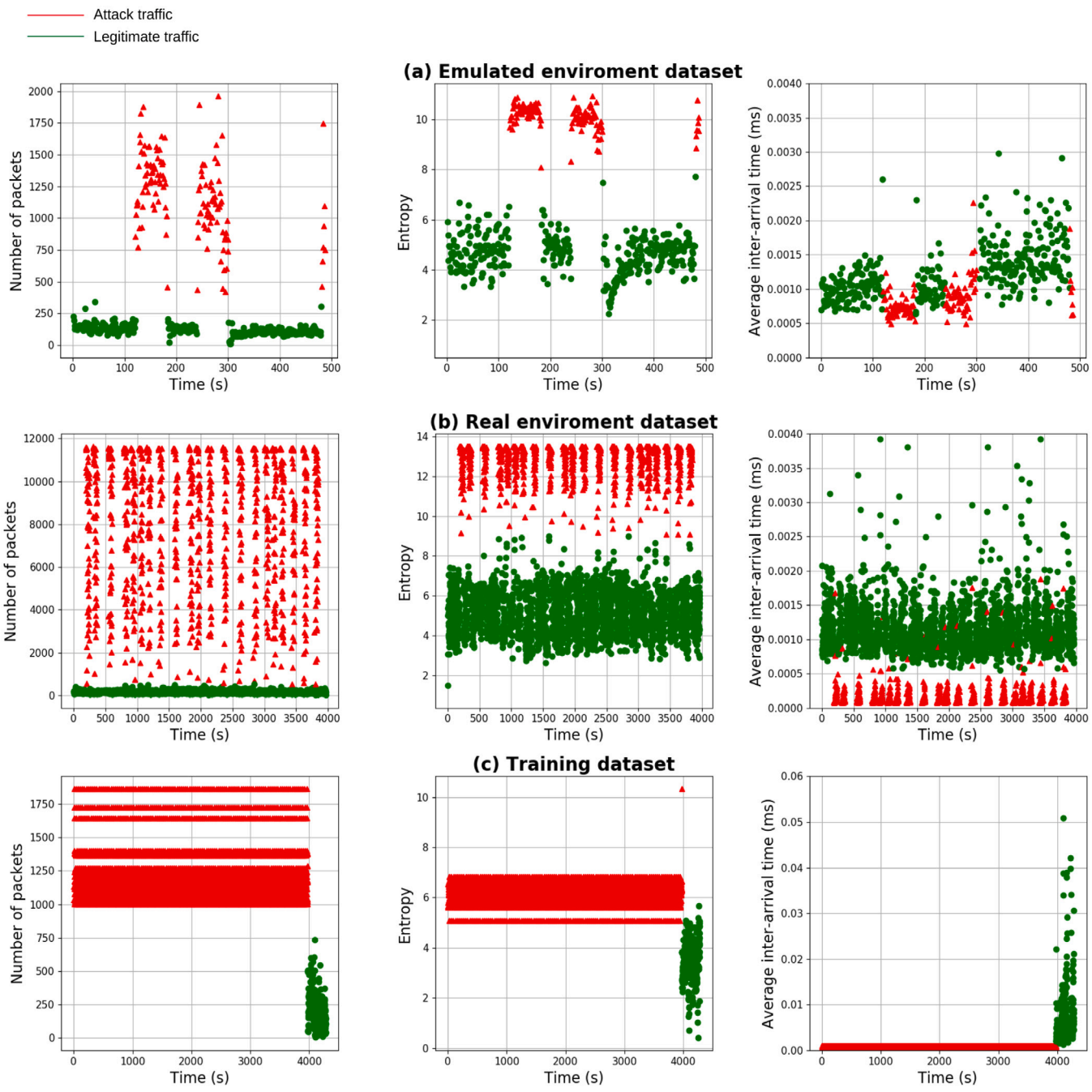

Fig. 9. Features in the (a) Emulated, (b) Real and (c) Training datasets for a time window with length $\Delta T = 1$ s.

trace obtained in the real environment has a size of 11.3 GB. For each of these two traces, a dataset is built with the values of the three features (number of packets, entropy and average of inter-arrival time) for each time window with length ΔT, as described in Section 4.1. The values of the three features for each in these two datasets are shown in Fig. 9(a) and (b). As can be seen, the number of packets and entropy are large for attack traffic and small for legitimate traffic, whereas the average inter-arrival time is smaller for attack traffic than for legitimate traffic.

The other two traces are the CAIDA [99] traffic trace, with a size of 427.6 MB used by Xiang et al. in [100] and a webserver traffic trace obtained at Instituto Federal de Educação, Ciência e Tecnologia do Tocantins (IFTO), consisting only of legitimate traffic with a size of 142.6 MB. Using firstly the CAIDA trace followed by the IFTO trace, we build only one dataset, called training dataset, which is used for the training phase of the machine learning algorithms described in previous section. Fig. 9(c) shows the values of the three features in the training dataset.

5.2. Emulated environment

In this scenario, we used the equipment and software listed in Table 3 with the intent of reproducing an environment close to the real one. The software chosen to reproduce the testbed was netkit [101] since it emulates the components of real machines and the Linux operating system with many of its features. The whole process is automated and managed by shell scripts developed using the netcat tools for connecting all virtual computers. To produce the emulated trace, the attack traffic is sent by the M-RoQ attack software and the legitimate

Table 3
Software and hardware specification for emulated testbed.

Computer host software	
Linux	Ubuntu — version 16.04 LTS — Xenial Xerus
Netkit-ng	Core — version 3.0.4
	Filesystem — version 7.0
	Kernel — version 3.2
	Linux — version 3.2.54-netkit-ng-K3.2
Computer host hardware	
Desktop	Memory — 3 GB
	Hard disk — 100 GB
Netkit software	
Hping3	version 3.0.0-alpha-2
Shell bash	version 4.2.37
Netcat (nc6)	version 1.0
IPTraf	version 3.0.0
TCPDUMP	version 4.3.0
Tshark	version 1.12.1
Slowhttptest	version 1.7
Apache2	version 2.4.33
Curl-loader	version 0.56
Netkit hardware	
Legitimate clients	memory — 100 MB
Attackers	memory — 128 MB
Routers 1–2	memory — 100 MB
Border router	memory — 256 MB
Victim	memory — 512 MB

traffic is generated by curl-loader [102]. Tcpdump and tshark [103] are used to collect the traffic data.

The process illustrated in Fig. 10 uses nine computers and ten routers to simulate a real Internet environment. To produce the trace, the legitimate client computers, Client 1 and Client 2, generate HTTP traffic while Client 3 generates FTP traffic, both using curl-loader. The Manager machine starts and stops the entire process. The attacking computers use the attack period of time (T) set to 1 s. The curl-loader is used to generate legitimate traffic in order to simulate the real traffic data of the Internet to test the link capacity of the server. The computers of Client 1 and Client 2 are configured to simulate 50 machines, each generating HTTP traffic. The computer of Client 3 is configured to simulate 20 machines generating FTP traffic, making a total of 120 machines transmitting traffic. The attack traffic consists of UDP packets each having a size of 1024 kbytes. The attack target computer is a Web server running Apache software.

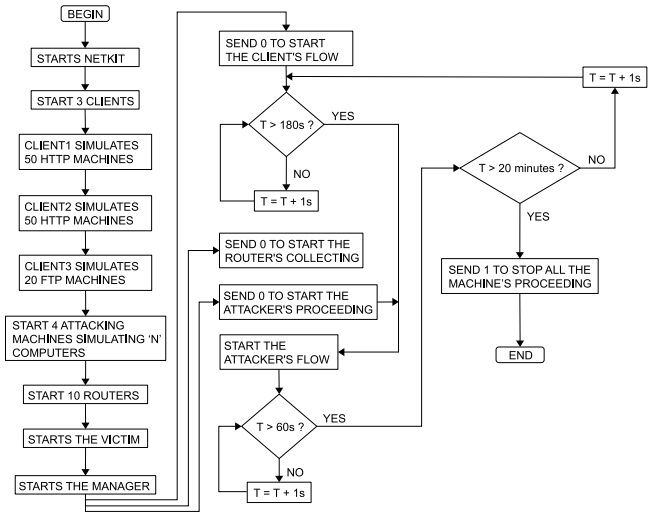

Fig. 10. Flowchart to produce the trace in the emulated environment.

Netkit by default starts the emulated machines with 32 MB of memory, but we set the amount of memory for each emulated machine as summarized in Table 3 in order to support other software that may be installed later, such as gcc, IPTraf and scripts created for this purpose. The target computer and the border router have more memory than the others because they spend more resources collecting data and processing all flows. The attacking machines have slightly more memory, because the M-RoQ software increases the memory usage for performing the RoQ attack.

5.3. Real environment

This scenario is implemented at IFTO, which has a very broad network structure, using the equipment and software listed in Table 4.

Table 4
Software and hardware specification for real testbed.

Software	
Linux Ubuntu	version 16.04 LTS — Xenial Xerus
Hping3	version 3.0.0-alpha-2
Shell bash	version 4.3.48
Netcat	version 1.105-7ubuntu1
IPTraf	version 3.0.0
TCPDUMP	version 4.9.0
Tshark	version 2.2.6
Slowhttptest	version 1.7
Apache2	version 2.4.33
Curl-loader	version 0.56
Cisco 2801 IOS	version 15.0
Hardware	
Desktop	Memory — 8 GB
	Hard disk — 1 TB
Router Cisco 2801 series	Memory — 128 MB
	FLASH memory — 32 MB

The network structure contains several routers among the classrooms, allowing data traffic from different locations thus mimicking the real Internet traffic. This environment has the same software suite used in the emulated environment with some additions and it is composed of five computers, one server and three Cisco routers.

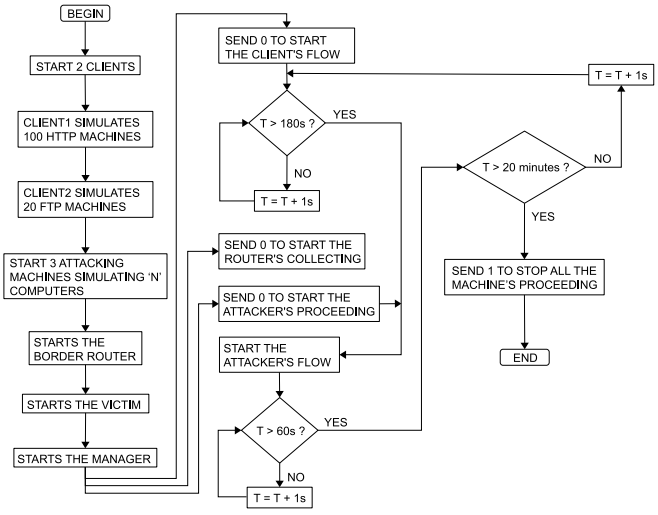

Fig. 11. Flowchart to produce the traffic trace in the real environment.

The process illustrated in Fig. 11 simulates a real Internet environment in a real testbed, where the legitimate Client 1 computer generates HTTP traffic while Client 2 generates FTP traffic, both using curl-loader. The attacking computers also used M-RoQ attack software with T = 1 s. Thus, the legitimate computer Client 1 was configured to simulate 100 machines generating HTTP traffic each and Client 2 was configured to simulate 20 machines, each one generating FTP traffic.

5.4. M-RoQ software

We developed M-RoQ (Manipulated-RoQ) attack software to generate RoQ-like attacks. To illustrate its operation, the scenario depicted

in Fig. 12 was implemented in an emulated environment using netkit software and is composed of 10 virtual Linux computers, of which five emulate one zombie machine each, generating an attack traffic towards the target (victim) represented by the computers marked in red color.

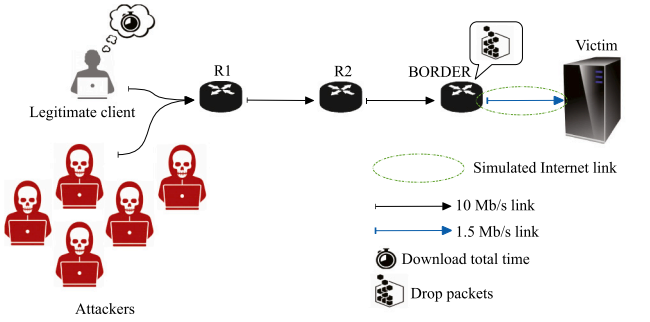

Fig. 12. M-RoQ software testbed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The process begins at the end customer machine. First of all, the attacking machine, the BORDER router and the victim computer start a server socket that wait for the signal of the client running on the legitimate client machine to initiate their activities. The server socket on the attacking machines will execute the RoQ attack using the Hping3 program after receiving the signal. Since Hping3 is a DDoS attack tool, it tries to completely stop the services of the victim. Thus, we developed a software in C language that manipulates Hping3 in a RoQ attack fashion. M-RoQ software consists of the Algorithm 1, that we made available online in [104]. The M-RoQ uses the Linux timeout command to create the ON time traffic and the usleep function to generate the OFF time period (which is the time gap between the former shockwave traffic attack and the next one). The attack flow is composite of UDP (-2 option) messages with spoofed random source IP addresses mimicking the public Internet IP.

Algorithm 1: M-RoQ software

```
time=0;
while time < 60 do
    system("timeout 0.3 hping3 --rand-source
    --flood -2 <dst IP> -p <dst PORT> -d 1024 &");
    usleep(700 * 1000);
    time++;
end
system("killall -9 hping3");
```

The server socket in the BORDER router collects the total amount of dropped packets as well as the flows arriving on it by using the tcpdump tool. In the victim, the server socket starts the IPTraf which measures the TCP throughput. Finally, the end customer (legitimate client) client socket signals all machines and runs a shell script which collects the total download time of the FTP. All the information collected during the experiments with the attack scenario is compared to the experiments without the attack scenario to verify if the M-RoQ software is working as described and if the QoS of the victim services has been reduced.

6. Results and discussion

In this section, we present the results related to all scenarios in two distinct environments as well as the M-RoQ software.

6.1. Performance metrics

To evaluate the performance of the classifiers, we use precision, recall, F1-score and confusion matrix table metrics, which are defined as follow [105]:

$$Precision = \frac{TP}{TP + FP}, \tag{6}$$

$$Recall = \frac{TP}{TP + FN}, \tag{7}$$

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{8}$$

where TP (True Positive) is the number of sample cases classified correctly, i.e., classifying attack traffic as an attack. The TN (True Negative) is the number of sample cases classified correctly, i.e., classifying legitimate traffic as legitimate. The FP (False Positive) is the number of positive sample cases incorrectly classified, i.e., classifying attack traffic as legitimate traffic and finally, FN (False Negative) is the number of negative sample cases incorrectly classified, i.e., classifying legitimate traffic as an attack. The F1-score metric is the harmonic mean between precision and recall. Since this measure is an average, it gives a more accurate view of the efficiency of the classifier than merely precision or recall. Therefore, these metrics are used for evaluating the performance of the proposed approaches to classify the advanced attacks considered in this work. The confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It allows the visualization of the classifier performance.

6.2. Impact of RoQ attack period on quality of service

In this subsection, we investigate the impact of the attack period on the quality of service (QoS) of the target (victim) using the M-RoQ software. Therefore, we define three scenarios with different attack periods T and one scenario without attack. Since a RoQ attack does not have the same attack characteristic than a Shrew attack, i.e., it does not have to try to identify the RTO (Retransmission TimeOut) of the TCP flow, we use three different attack periods with T = 1 s, T = 2 s, and T = 3 s, in order to investigate their impact on the reduction of the QoS of the victim. For each attack scenario, we consider a set of 100 experiments which consists in running 100 times the M-RoQ software plus 100 downloads of a file. For the scenario without attack, the set of 100 experiments consists only in 100 downloads of a file.

Table 5
QoS parameters with and without RoQ attacks for attack periods of T = 1 s, T = 2 s, and T = 3 s. D. Time: Download time.

T = 1 s	Mean	Max	Min	Standard Deviation	Margin of error	Confidence interval
Throughput (kb/s)	27.65	52.57	7.81	7.77	0.12	27.53 to 27.77
Delay (ms)	34.14	107.6	16.92	11.55	0.17	34.01 to 34.35
Jitter (ms)	63.08	124.91	33.84	13.66	0.16	62.91 to 63.24
Drop	3198.20	9676	240	1839.61	1.09	3197.11 to 3199.29
D. Time (s)	136.37	432	64	48.40	0.40	135.97 to 136.77
T = 2 s	**Mean**	**Max**	**Min**	**Standard Deviation**	**Margin of error**	**Confidence interval**
Throughput (kb/s)	79.39	110.72	54.10	9.02	0.10	79.30 to 79.49
Delay (ms)	12.16	17.39	8.75	1.34	0.02	12.14 to 12.18
Jitter (ms)	24.21	31.30	17.50	2.46	0.04	24.18 to 24.25
Drop	572.39	1628	30	331.79	0.97	571.42 to 573.36
D. Time (s)	42.52	60	31	4.68	0.06	42.46 to 42.58
T = 3 s	**Mean**	**Max**	**Min**	**Standard Deviation**	**Margin of error**	**Confidence interval**
Throughput (kb/s)	113.89	124.88	94.53	7.28	0.07	113.83 to 113.96
Delay (ms)	8.53	10.24	7.52	0.60	0.01	8.52 to 8.54
Jitter (ms)	17.03	20.48	15.04	1.16	0.02	17.02 to 17.05
Drop	396.08	1807	2	306.92	1.21	394.87 to 397.29
D. Time (s)	29.39	36	27	2.08	0.03	29.36 to 29.42
Without attack	**Mean**	**Max**	**Min**	**Standard Deviation**	**Margin of error**	**Confidence interval**
Throughput (kb/s)	176.61	187.88	168.11	3.03	0.02	176.59 to 176.63
Delay (ms)	5.57	5.78	5.46	0.15	0	5.57 to 5.58
Jitter (ms)	11.15	11.55	10.93	0.29	0.01	11.14 to 11.15
Drop	0	0	0	0	0	0
D. Time (s)	18.92	20	18	0.31	0.01	18.91 to 18.93

For each collected parameter, we present the mean, the maximum, the minimum, the standard deviation and the margin of error values

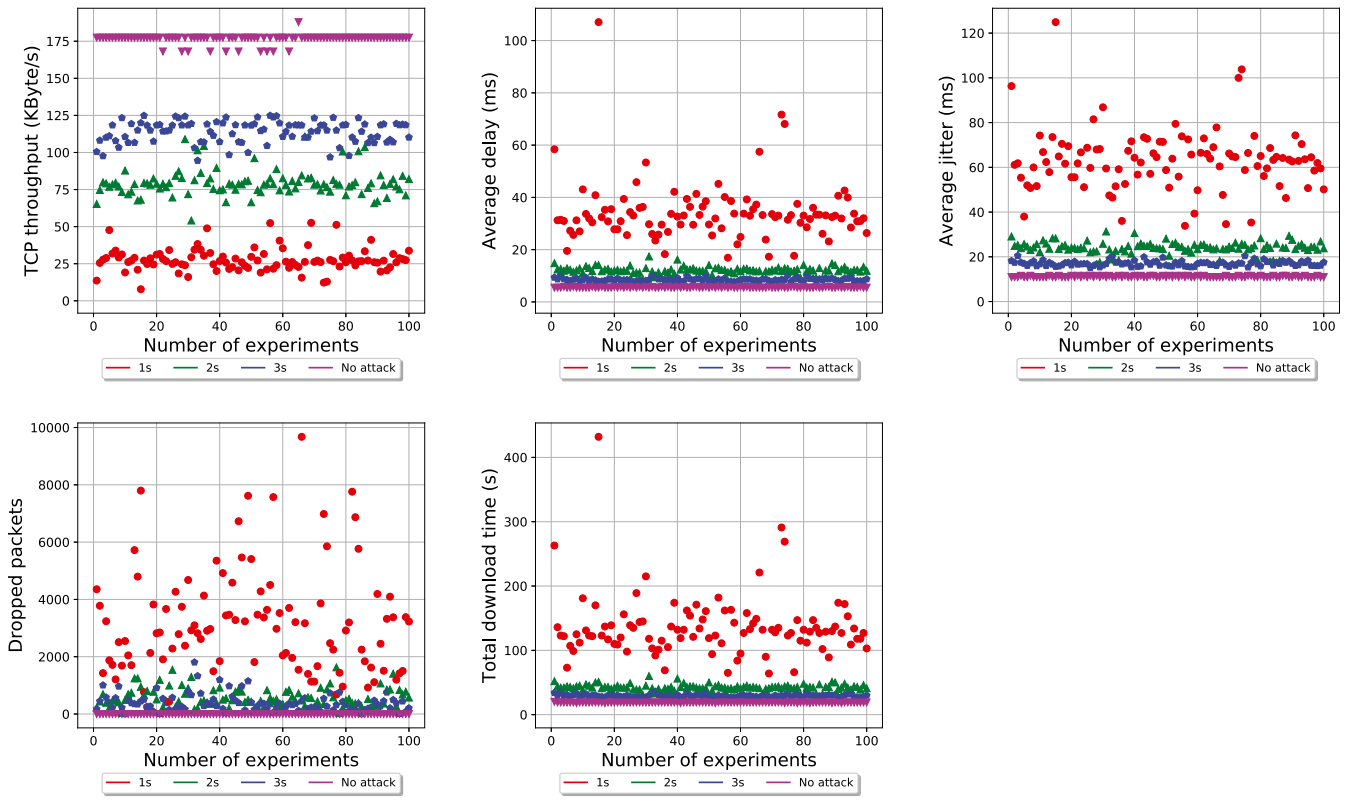

Fig. 13. QoS parameters as a function of the number of experiments with and without RoQ attacks for attack periods of T = 1 s, T = 2 s, and T = 3 s. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

based on a confidence interval of 95% as summarized in Table 5. The results depicted in Fig. 13 show that the QoS of the victim is compromised for the three attack time periods. As can be seen in this figure, all the graphics with purple color have the best QoS for the applications because there is no attack traffic, while the QoS for the other colors are decreasing as the period of time T decreases. The attack with T = 1 s leads to the larger reduction of the QoS because it is the shortest period of time before the next shockwave traffic flood. Thus, the longer the period of time before the next shockwave traffic flood, the faster will be the recovery of the end system protocols in adapting their sending rates.

Additionally, the TCP protocol is more impacted in the experiment with T = 1 s because when it tries to recover from the packet loss after the RTO, it will more quickly face another shockwave traffic flood forcing it to use a new RTO.

6.3. Performance of the classification approaches

This section provides an evaluation of the classification approaches for the traffic traces obtained in emulated and real environments. In Tables 6–8, we provide the evaluation of the four machine learning algorithms using each feature separately: number of packets, average inter-arrival time and entropy, respectively.

In Table 9, we provide the evaluation of the machine learning algorithms using the three features, i.e. number of packets, entropy and average of inter-arrival times, and in Table 10, we provide the results of the evaluation of the proposed approach based on FL, MLP and ED.

As can be seen in Tables 6–9, the choice of the features can deeply impact the performance of the classification. As we can see in Tables 6–8, the separate use of each of the three features leads to poor classification results. This is due to the fact that the features have values that are, at some points, very close to each other, causing the algorithm to incorrectly classify the data for that feature. In this case, the classifier with the joint use of the three features leads to a better performance for detecting attacks than the classifier with one feature. This can also be seen in the confusion matrix column in Tables 6–9, where the amount of FP and FN in Tables 6–8 is larger than the amount of FP and FN in Table 9.

As we can see in Table 9, the results of the classification for emulated traffic show that precision ranges from 94.53% (MNB) to 100% (MLP, K-NN, SVM) for attack traffic and from 94.69% (SVM) to 98.62% (MLP) for legitimate traffic and recall ranges from 84.62% (SVM) to 96.15% (MLP) for attack traffic and from 98.04% (MNB) to 99.30% (MLP, K-NN, SVM) for legitimate traffic. The results of the classification for real traffic show that precision ranges from 92.36% (MNB) to 100% (MLP, K-NN, SVM) for attack traffic and from 99.57% (SVM) to 99.89% (MLP) for legitimate traffic and recall ranges from 98.99% (SVM) to 99.75% (MLP) for attack traffic and from 96.72% (MNB) to 100% (MLP, K-NN, SVM) for legitimate traffic.

We also show in Table 9 that F1-score ranges from 91.67% (SVM) to 98.04% (MLP) for attack traffic and from 97.28% (SVM) to 99.30% (MLP) for legitimate traffic, while, for real traffic, F1-score ranges from 96.02% (MNB) to 99.87% (MLP) for attack traffic and from 98.21% (MNB) to 99.95% (MLP) for legitimate traffic. According to Table 9, MLP leads to a slightly better performance than the other three machine learning algorithms under study.

Table 10 shows the results of the performance evaluation for the proposed approach based on FL, MLP and ED. As we can see, it leads to a better performance than the four machine learning algorithms as shown in Table 9 due to the fact that the warning traffic classification leads to a decrease of false negative and false positive rates. This can be observed in Table 11. The traffic marked as warning has the feature *Number of packets* smaller than 1000 packets, which is not considered as an attack, but it has a large value for the entropy, which can be considered as an attack. Therefore, this abnormality can degrade the performance of the classifier. This abnormality can be caused by two factors. The first factor occurs when we apply the sliding time window with length of 1 s over the traffic trace and the last packets of an attack

Table 6
Results of the performance evaluation of the four machine learning algorithms using number of packets for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$		
Emulated	K-NN						**attack**	**legitimate**
		attack	100.00%	88.46%	93.88%	**attack**	115	15
		legitimate	95.97%	100.00%	97.94%	**legitimate**	0	357
	SVM						**attack**	**legitimate**
		attack	100.00%	2.31%	4.51%	**attack**	3	127
		legitimate	73.76%	100.00%	84.90%	**legitimate**	0	357
	MNB						**attack**	**legitimate**
		attack	26.69%	100.00%	42.14%	**attack**	130	0
		legitimate	0%	0%	0%	**legitimate**	357	0
	MLP						**attack**	**legitimate**
		attack	26.69%	100.00%	42.14%	**attack**	130	0
		legitimate	0%	0%	0%	**legitimate**	357	0
Real	K-NN						**attack**	**legitimate**
		attack	100.00%	99.16%	99.58%	**attack**	1179	10
		legitimate	99.64%	100.00%	99.82%	**legitimate**	0	2777
	SVM						**attack**	**legitimate**
		attack	100.00%	0.17%	0.34%	**attack**	2	1187
		legitimate	70.06%	100.00%	82.39%	**legitimate**	0	2777
	MNB						**attack**	**legitimate**
		attack	29.98%	100.00%	46.13%	**attack**	1187	0
		legitimate	0%	0%	0%	**legitimate**	2777	0
	MLP						**attack**	**legitimate**
		attack	29.98%	100.00%	46.13%	**attack**	1187	0
		legitimate	0%	0%	0%	**legitimate**	2777	0

Table 7
Results of the performance evaluation of the four machine learning algorithms using average inter-arrival time for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$		
Emulated	K-NN						**attack**	**legitimate**
		attack	36.21%	96.92%	52.72%	**attack**	126	4
		legitimate	97.12%	37.82%	54.44%	**legitimate**	222	135
	SVM						**attack**	**legitimate**
		attack	26.69%	100%	42.14%	**attack**	130	0
		legitimate	0%	0%	0%	**legitimate**	357	0
	MNB						**attack**	**legitimate**
		attack	26.69%	100.00%	42.14%	**attack**	130	0
		legitimate	0%	0%	0%	**legitimate**	357	0
	MLP						**attack**	**legitimate**
		attack	26.69%	100.00%	42.14%	**attack**	130	0
		legitimate	0%	0%	0%	**legitimate**	357	0
Real	K-NN						**attack**	**legitimate**
		attack	36.72%	99.33%	53.62%	**attack**	1181	8
		legitimate	98.93%	26.72%	42.08%	**legitimate**	2035	742
	SVM						**attack**	**legitimate**
		attack	29.98%	100.00%	46.13%	**attack**	1189	0
		legitimate	0%	0%	0%	**legitimate**	2777	0
	MNB						**attack**	**legitimate**
		attack	29.98%	100.00%	46.13%	**attack**	1187	0
		legitimate	0%	0%	0%	**legitimate**	2777	0
	MLP						**attack**	**legitimate**
		attack	29.98%	100.00%	46.13%	**attack**	1187	0
		legitimate	0%	0%	0%	**legitimate**	2777	0

Table 8
Results of the performance evaluation of the four machine learning algorithms using entropy for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$		
Emulated	K-NN						**attack**	**legitimate**
		attack	56.03%	100.00%	71.82%	**attack**	130	0
		legitimate	100.00%	71.43%	83.33%	**legitimate**	102	255
	SVM						**attack**	**legitimate**
		attack	48.50%	99.23%	65.15%	**attack**	129	1
		legitimate	99.55%	61.62%	76.12%	**legitimate**	137	220
	MNB						**attack**	**legitimate**
		attack	26.69%	100.00%	42.14%	**attack**	130	0
		legitimate	0%	0%	0%	**legitimate**	357	0
	MLP						**attack**	**legitimate**
		attack	26.69%	100.00%	42.14%	**attack**	130	0
		legitimate	0%	0%	0%	**legitimate**	357	0
Real	K-NN						**attack**	**legitimate**
		attack	46.83%	100.00%	63.79%	**attack**	1189	0
		legitimate	100.00%	51.39%	67.89%	**legitimate**	1350	1427
	SVM						**attack**	**legitimate**
		attack	8.88%	12.20%	10.28%	**attack**	145	1044
		legitimate	55.25%	46.42%	50.45%	**legitimate**	1488	1289
	MNB						**attack**	**legitimate**
		attack	29.98%	100.00%	46.13%	**attack**	1187	0
		legitimate	0%	0%	0%	**legitimate**	2777	0
	MLP						**attack**	**legitimate**
		attack	29.98%	100.00%	46.13%	**attack**	1187	0
		legitimate	0%	0%	0%	**legitimate**	2777	0

Table 9
Results of the performance evaluation of the four machine learning algorithms using three features for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$		
Emulated	MLP						**attack**	**legitimate**
		attack	100.00%	96.15%	98.04%	**attack**	125	5
		legitimate	98.62%	100.00%	99.30%	**legitimate**	0	357
	K-NN						**attack**	**legitimate**
		attack	100.00%	88.46%	93.88%	**attack**	115	15
		legitimate	95.97%	100.00%	97.94%	**legitimate**	0	357
	SVM						**attack**	**legitimate**
		attack	100.00%	84.62%	91.67%	**attack**	110	20
		legitimate	94.69%	100.00%	97.28%	**legitimate**	0	357
	MNB						**attack**	**legitimate**
		attack	94.53%	93.08%	93.80%	**attack**	121	9
		legitimate	97.49%	98.04%	97.77%	**legitimate**	7	350
Real	MLP						**attack**	**legitimate**
		attack	100.00%	99.75%	99.87%	**attack**	1186	3
		legitimate	99.89%	100.00%	99.95%	**legitimate**	0	2777
	K-NN						**attack**	**legitimate**
		attack	100.00%	99.16%	99.58%	**attack**	1179	10
		legitimate	99.64%	100.00%	99.82%	**legitimate**	0	2777
	SVM						**attack**	**legitimate**
		attack	100.00%	98.99%	99.49%	**attack**	1177	12
		legitimate	99.57%	100.00%	99.78%	**legitimate**	0	2777
	MNB						**attack**	**legitimate**
		attack	92.85%	99.41%	96.02%	**attack**	1182	7
		legitimate	99.74%	96.72%	98.21%	**legitimate**	91	2686

Table 10
Results of the performance evaluation of the proposed approach based on FL, MLP and ED for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$		
							attack	**legitimate**
Emulated	FL, MLP and ED	attack	100.00%	97.70%	98.80%	**attack**	114	3
		legitimate	99.20%	100.00%	99.60%	**legitimate**	0	357
							attack	**legitimate**
Real	FL, MLP and ED	attack	100.00%	100.00%	100.00%	**attack**	1187	0
		legitimate	100.00%	100.00%	100.00%	**legitimate**	0	2762

are included in the beginning of a time window, being composed mostly by legitimate traffic, which may lead to an increase of the entropy in that time window due to the presence of some packets of the attack. The second factor can occur when an attack is in its final phase, where the attack power is weaker and the last malicious packets may be included in a time window composed mostly by legitimate traffic, which may lead to an increase of the entropy in that time window.

Table 11
Warning alerts.

	Attributes/Values	ED classification	FL classification	MLP classification
Emulated	Number of packets — 422 Entropy — 8.72	normal attack	attack	normal
	Number of packets — 464 Entropy — 8.84	normal attack	attack	normal
Real	Number of packets — 461 Entropy — 8.84	normal attack	attack	normal
	Number of packets — 390 Entropy — 8.60	normal attack	attack	normal
	Number of packets — 348 Entropy — 8.43	normal attack	attack	normal
	Number of packets — 453 Entropy — 8.82	normal attack	attack	normal
	Number of packets — 325 Entropy — 8.34	normal attack	attack	normal
	Number of packets — 321 Entropy — 8.32	normal attack	attack	normal
	Number of packets — 482 Entropy — 8.91	normal attack	attack	normal
	Number of packets — 350 Entropy — 8.45	normal attack	attack	normal
	Number of packets — 437 Entropy — 8.76	normal attack	attack	normal
	Number of packets — 502 Entropy — 8.97	normal attack	attack	normal
	Number of packets — 513 Entropy — 8.99	normal attack	attack	normal
	Number of packets — 322 Entropy — 8.33	normal attack	attack	normal
	Number of packets — 399 Entropy — 8.64	normal attack	attack	normal
	Number of packets — 538 Entropy — 9.06	normal attack	attack	normal
	Number of packets — 461 Entropy — 8.84	normal attack	attack	normal
	Number of packets — 386 Entropy — 8.59	normal attack	attack	normal
	Number of packets — 336 Entropy — 8.39	normal attack	attack	normal
	Number of packets — 540 Entropy — 9.07	normal attack	attack	normal

As we can see in Table 10, when using the approach based on FL, MLP and ED, for classification of emulated traffic, we obtained a precision of 100% for attack traffic and 99.20% for legitimate traffic, a recall of 97.70% for attack traffic and 100% for legitimate traffic, and a F1-score of 98.80% for attack traffic and 99.60% for legitimate traffic, while, for real traffic, we obtained a precision of 100% for attack traffic and 100% for legitimate traffic, a recall of 100% for attack traffic and 100% for legitimate traffic and a F1-score of 100% for attack traffic and 100% for legitimate traffic.

6.4. Resource usage

The computational resources used by the four machine learning algorithms and the approach based on FL, MLP and ED were collected by a computer running the GNU/Linux operating system Ubuntu version 19.04 (Disco Dingo), with 8 GB of RAM, a 64-bit AMD Phenom II X4 925 processor with 4 cores and 1 thread per core.

Table 12
Computational resource usage.

Algorithm	Emulated dataset	Real dataset
K-NN	CPU: 122% RAM: 0.76 MB Execution time: 0.67 ms	CPU: 119% RAM: 0.77 MB Execution time: 0.79 ms
MLP	CPU: 117% RAM: 0.76 MB Execution time: 0.74 ms	CPU: 116% RAM: 0.77 MB Execution time: 0.87 ms
SVM	CPU: 117% RAM: 0.77 MB Execution time: 0.66 ms	CPU: 117% RAM: 0.76 MB Execution time: 0.79 ms
MNB	CPU: 120% RAM: 0.77 MB Execution time: 0.66 ms	CPU: 118% RAM: 0.70 MB Execution time: 0.74 ms
FL, MLP and ED	CPU: 125% RAM: 6.6 MB Execution time: 11:46.49 min	CPU: 125% RAM: 5.79 MB Execution time: 46:48.44 min

Table 12 shows the resource usage by each classification approach. CPU consumption and execution times were collected through the GNU/Linux command /usr/bin/time. RAM usage was collected using the psrecord [106] software. Since CPU consumption is larger than 100% it means that at least 2 cores were used to classify the whole dataset. As can be seen in this table, the better performance of the proposed approach based on FL, MLP and ED has a cost in terms of the usage of computational resources: it requires 6.6 MB and 5.79 MB of RAM for the emulated and real traffic datasets, respectively, while machine learning algorithms require less or equal to 0.77 MB of RAM for both datasets and it requires 11:46.49 min and 46:48.44 min to finish the classification for the emulated and real traffic datasets, respectively, while machine learning algorithms require less than 0.9 ms for both datasets. Nevertheless, the code used for implementing the approach based on FL, MLP and ED was not optimized, being this task left for further work.

7. Conclusion

In this paper, we evaluated and compared four machine learning algorithms for the detection of RoQ attacks: MLP, K-NN, SVM and MNB. We also proposed an approach based on a combination of three methods, FL, MLP and ED, for the detection of RoQ attacks. We evaluated these approaches using emulated and real traffic traces. To build the traffic traces, we created an emulated environment and a real environment and developed an attack tool to generate the attacks, called M-RoQ. We showed that the use of three features, namely number of packets, entropy and average inter-arrival time, leads to a better classification of the four machine learning algorithms than using only the entropy as a feature. We showed that, among the four machine learning algorithms, MLP leads to the best classification results on the detection of RoQ attacks and that the approach based on FL, MLP and ED outperforms MLP at the cost of a larger execution time. For future work, we plan to explore other machine learning algorithms and compare the results with other approaches.

CRediT authorship contribution statement

Vinícius de Miranda Rios: Conceptualization, Methodology, Investigation, Software, Validation, Formal analysis, Data curation, Visualization, Writing - original draft, Writing - review & editing. **Pedro R.M. Inácio:** Conceptualization, Methodology, Investigation, Validation, Writing - review & editing, Project administration, Funding acquisition. **Damien Magoni:** Conceptualization, Methodology, Investigation, Validation, Writing - review & editing, Supervision, Project administration. **Mário M. Freire:** Conceptualization, Methodology, Investigation, Validation, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C. Townsley, Are businesses getting complacent when it comes to DDoS mitigation? Netw. Secur. 2018 (6) (2018) 6–9.

[2] A. Chadd, DDoS attacks: Past, present and future, Netw. Secur. 2018 (7) (2018) 13–15.

[3] S. Newman, Under the radar: The danger of stealthy DDoS attacks, Netw. Secur. 2019 (2) (2019) 18–19.

[4] Akamai, [State of the internet]/security a year in review, 2018, URL: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/2018-state-of-the-internet-security-a-year-in-review.pdf. (Accessed 13 December 2018).

[5] G. Somani, M.S. Gaur, D. Sanghi, M. Conti, M. Rajarajan, Scale inside-out: Rapid mitigation of cloud DDoS attacks, IEEE Trans. Dependable Secure Comput. 15 (6) (2017) 959–973, http://dx.doi.org/10.1109/TDSC.2017.2763160.

[6] A. Wang, W. Chang, S. Chen, A. Mohaisen, A data-driven study of DDoS attacks and their dynamics, IEEE Trans. Dependable Secure Comput. 14 (8) (2015) http://dx.doi.org/10.1109/TDSC.2018.2808344.

[7] T. Peng, C. Leckie, K. Ramamohanarao, Survey of network-based defense mechanisms countering the DoS and DDoS problems, ACM Comput. Surv. 39 (1) (2007) 3, http://dx.doi.org/10.1145/1216370.1216373.

[8] R.K.C. Chang, Defending against flooding-based distributed denial-of-service attacks: A tutorial, IEEE Commun. Mag. 40 (10) (2002) 42–51, http://dx.doi.org/10.1109/MCOM.2002.1039856.

[9] A. Saied, R.E. Overill, T. Radzik, Detection of known and unknown DDoS attacks using artificial neural networks, Neurocomputing 172 (2016) 385–393, http://dx.doi.org/10.1016/j.neucom.2015.04.101.

[10] C. Douligeris, A. Mitrokotsa, DDoS attacks and defense mechanisms: Classification and state-of-the-art, Comput. Netw. 44 (5) (2004) 643–666, http://dx.doi.org/10.1016/j.comnet.2003.10.003.

[11] B. Harris, R. Hunt, TCP/IP security threats and attack methods, Comput. Commun. 22 (10) (1999) 885–897, http://dx.doi.org/10.1016/S0140-3664(99)00064-X.

[12] X. Ma, B. An, M. Zhao, X. Luo, L. Xue, Z. Li, T. Miu, X. Guan, Randomized security patrolling for link flooding attack detection, IEEE Trans. Dependable Secure Comput. (2019) http://dx.doi.org/10.1109/TDSC.2019.2892370.

[13] K.J. Singh, T. De, MLP-GA Based algorithm to detect application layer DDoS attack, J. Inform. Secur. Appl. 36 (2017) 145–153, http://dx.doi.org/10.1016/j.jisa.2017.09.004.

[14] N. Tripathi, N. Hubballi, Slow rate denial of service attacks against HTTP/2 and detection, Comput. Secur. 52 (2018) 255–272, http://dx.doi.org/10.1016/j.cose.2017.09.009.

[15] H. D'Cruze, P. Wang, R.O. Sbeit, A. Ray, A software-defined networking (SDN) approach to mitigating DDoS attacks, in: Information Technology-New Generations, Springer, 2018, pp. 141–145, http://dx.doi.org/10.1007/978-3-319-54978-1_19.

[16] O. Osanaiye, K.-K.R. Choo, M. Dlodlo, Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework, J. Netw. Comput. Appl. 67 (2016) 147–165, http://dx.doi.org/10.1016/j.jnca.2016.01.001.

[17] J. Mirkovic, P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53, http://dx.doi.org/10.1145/997150.997156.

[18] B.A. Khalaf, S.A. Mostafa, A. Mustapha, M.A. Mohammed, W.M. Abduallah, Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods, IEEE Access 7 (2019) 51691–51713, http://dx.doi.org/10.1109/ACCESS.2019.2908998.

[19] S. Behal, K. Kumar, M. Sachdeva, Characterizing DDoS attacks and flash events: Review, research gaps and future directions, Comp. Sci. Rev. 25 (2017) 101–114, http://dx.doi.org/10.1016/j.cosrev.2017.07.003.

[20] M. Aamir, M.A. Zaidi, A survey on DDoS attack and defense strategies: From traditional schemes to current techniques, Interdiscip. Inform. Sci. 19 (2) (2013) 173–200, http://dx.doi.org/10.4036/iis.2013.173.

[21] G. Somani, M.S. Gaur, D. Sanghi, M. Conti, R. Buyya, DDoS attacks in cloud computing: Issues, taxonomy, and future directions, Comput. Commun. 107 (2017) 30–48, http://dx.doi.org/10.1016/j.comcom.2017.03.010.

[22] N. Agrawal, S. Tapaswi, Defense schemes for variants of distributed denial-of-service (DDoS) attacks in cloud computing: A survey, Inf. Secur. J. Glob. Perspect. 26 (2) (2017) 61–73, http://dx.doi.org/10.1080/19393555.2017.1282995.

[23] N.S. Rao, K.C. Sekharaiah, A.A. Rao, A survey of distributed denial-of-service (DDoS) defense techniques in ISP domains, in: Innovations in Computer Science and Engineering, Springer, 2019, pp. 221–230, http://dx.doi.org/10.1007/978-981-10-8201-6_25.

[24] A.P. Abidoye, I.C. Obagbuwa, DDoS attacks in WSNs: Detection and countermeasures, IET Wirel. Sens. Syst. 8 (2) (2017) 52–59, http://dx.doi.org/10.1049/iet-wss.2017.0029.

[25] H.H.R. Sherazi, R. Iqbal, F. Ahmad, Z.A. Khan, M.H. Chaudhary, DDoS attack detection: A key enabler for sustainable communication in internet of vehicles, Sustain. Comput. Inform. Syst. 23 (2019) 13–20, http://dx.doi.org/10.1016/j.suscom.2019.05.002.

[26] L. Zhou, H. Guo, G. Deng, A fog computing based approach to DDoS mitigation in IIoT systems, Comput. Secur. 85 (2019) 51–62, http://dx.doi.org/10.1016/j.cose.2019.04.017.

[27] J. Cui, M. Wang, Y. Luo, H. Zhong, DDoS detection and defense mechanism based on cognitive-inspired computing in SDN, Future Gener. Comput. Syst. 97 (2019) 275–283, http://dx.doi.org/10.1016/j.future.2019.02.037.

[28] R. Sahay, G. Blanc, Z. Zhang, H. Debar, ArOMA: An SDN based autonomic DDoS mitigation framework, Comput. Secur. 70 (2017) 482–499, http://dx.doi.org/10.1016/j.cose.2017.07.008.

[29] R. Priyadarshini, R.K. Barik, A deep learning based intelligent framework to mitigate DDoS attack in fog environment, J. King Saud Univ., Comput. Inf. Sci. (2019) http://dx.doi.org/10.1016/j.jksuci.2019.04.010.

[30] A.E. Agoni, M. Dlodlo, Ip spoofing detection for preventing DDoS attack in fog computing, in: The 6th Global Wireless Summit, IEEE, 2018, pp. 43–46, http://dx.doi.org/10.1109/GWS.2018.8686626.

[31] DDoS Attacks, The largest DDoS attacks & what you can learn from them, 2019, URL: https://securityboulevard.com/2019/08/the-largest-ddos-attacks-what-you-can-learn-from-them/. (Accessed 01 September 2019).

[32] DDoS Attacks, DDoS attack that disrupted internet was largest of its kind in history, experts say, 2016, URL: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet. (Accessed 09 March 2019).

[33] DDoS Attacks, 5 notable DDoS attacks of 2017, 2017, URL: https://www.tripwire.com/state-of-security/featured/5-notable-ddos-attacks-2017/. (Accessed 14 July 2018).

[34] DDoS Attacks, Top 5 DDoS attacks of all times, 2019, URL: https://baltimorepostexaminer.com/top-5-ddos-attacks-of-all-times/2019/08/19. (Accessed 01 September 2019).

[35] DDoS Attacks, Wikipedia site paralyzed in several countries due to massive DDoS attack, 2019, URL: https://www.techtimes.com/articles/245274/20190908/wikipedia-site-paralyzed-in-several-countries-due-to-massive-ddos-attack.htm. (Accessed 10 September 2019).

[36] DDoS Attacks, Major DDoS attacks increased 967% this year, 2018, URL: https://ddosattacks.net/major-ddos-attacks-increased-967-this-year/. (Accessed 27 March 2019).

[37] DDoS Attacks, DDoS Attacks jump 18% YoY in Q2, 2019, URL: https://www.infosecurity-magazine.com/news/ddos-attacks-jump-18-yoy-in-q2/. (Accessed 15 August 2019).

[38] M. Guirguis, A. Bestavros, I. Matta, Exploiting the transients of adaptation for RoQ attacks on internet resources, in: 12th IEEE International Conference on Network Protocols, IEEE, 2004, pp. 184–195, http://dx.doi.org/10.1109/ICNP.2004.1348109.

[39] K. Wen, J. Yang, F. Cheng, C. Li, Z. Wang, H. Yin, Two-stage detection algorithm for RoQ attack based on localized periodicity analysis of traffic anomaly, in: 23rd International Conference on Computer Communication and Networks, IEEE, 2014, pp. 1–6, http://dx.doi.org/10.1109/ICCCN.2014.6911829.

[40] G. Yu, T. Li, J. Wei, C. Liu, Assessment of reduction of quality attacks on mobile IP networks, in: IEEE International Symposium on Parallel and Distributed Processing with Applications and IEEE International Conference on Ubiquitous Computing and Communications, IEEE, 2017, pp. 449–453, http://dx.doi.org/10.1109/ISPA/IUCC.2017.00073.

[41] A. Kuzmanovic, E.W. Knightly, Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2003, pp. 75–86, http://dx.doi.org/10.1145/863955.863966.

[42] C. Zhang, Z. Cai, W. Chen, X. Luo, J. Yin, Flow level detection and filtering of low-rate DDoS, Comput. Netw. 56 (15) (2012) 3417–3431, http://dx.doi.org/10.1016/j.comnet.2012.07.003.

[43] Z. Chen, C.K. Yeo, B.S. Lee, C.T. Lau, Power spectrum entropy based detection and mitigation of low-rate DoS attacks, Comput. Netw. 136 (2018) 80–94, http://dx.doi.org/10.1016/j.comnet.2018.02.029.

[44] J. Luo, X. Yang, The NewShrew attack: A new type of low-rate TCP-targeted DoS attack, in: Communication and Information Systems Security Symposium, IEEE, 2014, pp. 713–718, http://dx.doi.org/10.1109/ICC.2014.6883403.

[45] G. Maciá-Fernández, J.E. Díaz-Verdejo, P. García-Teodoro, Evaluation of a low-rate DoS attack against iterative servers, Comput. Netw. 51 (4) (2007) 1013–1030, http://dx.doi.org/10.1016/j.comnet.2006.07.002.

[46] G. Maciá-Fernández, J.E. Díaz-Verdejo, P. García-Teodoro, Mathematical model for low-rate DoS attacks against application servers, IEEE Trans. Inf. Forensics Secur. 4 (3) (2009) 519–529, http://dx.doi.org/10.1109/TIFS.2009.2024719.

[47] E. Damon, J. Dale, E. Laron, J. Mache, N. Land, R. Weiss, Hands-on denial of service lab exercises using SlowLoris and RUDY, in: Proceedings of the 2012 Information Security Curriculum Development Conference, ACM, 2012, pp. 21–29, http://dx.doi.org/10.1145/2390317.2390321.

[48] A. Sangodoyin, B. Modu, I. Awan, J.P. Disso, An approach to detecting distributed denial of service attacks in software defined networks, in: 6th International Conference on Future Internet of Things and Cloud, IEEE, 2018, pp. 436–443, http://dx.doi.org/10.1109/FiCloud.2018.00069.

[49] M. Aiello, E. Cambiaso, M. Mongelli, G. Papaleo, An on-line intrusion detection approach to identify low-rate DoS attacks, in: International Carnahan Conference on Security Technology, IEEE, 2014, pp. 1–6, http://dx.doi.org/10.1109/CCST.2014.6987039.

[50] S.T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, IEEE Commun. Surv. Tutor. 15 (4) (2013) 2046–2069, http://dx.doi.org/10.1109/SURV.2013.031413.00127.

[51] J. Park, K. Iwai, H. Tanaka, T. Kurokawa, Analysis of slow read DoS attack, in: International Symposium on Information Theory and Its Applications, IEEE, 2014, pp. 60–64.

[52] S. Shafieian, M. Zulkernine, A. Haque, CloudZombie: Launching and detecting slow-read distributed denial of service attacks from the cloud, in: International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, IEEE, 2015, pp. 1733–1740, http://dx.doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.261.

[53] E. Cambiaso, G. Chiola, M. Aiello, Introducing the SlowDrop attack, Comput. Netw. 150 (2019) 234–249, http://dx.doi.org/10.1016/j.comnet.2019.01.007.

[54] M. Aiello, G. Papaleo, E. Cambiaso, SlowReq: A weapon for cyberwarfare operations. Characteristics, limits, performance, remediations, in: International Joint Conference SOCO'13-CISIS'13-ICEUTE'13, Springer, 2014, pp. 537–546, http://dx.doi.org/10.1007/978-3-319-01854-6_55.

[55] E. Cambiaso, G. Papaleo, M. Aiello, Slowcomm: Design, development and performance evaluation of a new slow DoS attack, J. Inform. Secur. Appl. 35 (2017) 23–31, http://dx.doi.org/10.1016/j.jisa.2017.05.005.

[56] E. Cambiaso, G. Papaleo, G. Chiola, M. Aiello, Designing and modeling the slow next DoS attack, in: Computational Intelligence in Security for Information Systems Conference, Springer, 2015, pp. 249–259, http://dx.doi.org/10.1007/978-3-319-19713-5_22.

[57] S. Hosseini, M. Azizi, The hybrid technique for DDoS detection with supervised learning algorithms, Comput. Netw. 158 (2019) 35–45, http://dx.doi.org/10.1016/j.comnet.2019.04.027.

[58] N. Meti, D.G. Narayan, V.P. Baligar, Detection of distributed denial of service attacks using machine learning algorithms in software defined networks, in: International Conference on Advances in Computing, Communications and Informatics, IEEE, 2017, pp. 1366–1371, http://dx.doi.org/10.1109/ICACCI.2017.8126031.

[59] X. Liang, T. Znati, On the performance of intelligent techniques for intensive and stealthy ddos detection, Comput. Netw. 164 (2019) 106906, http://dx.doi.org/10.1016/j.comnet.2019.106906.

[60] M. Aamir, S.M.A. Zaidi, Clustering based semi-supervised machine learning for DDoS attack classification, J. King Saud Univ., Comput. Inf. Sci. (2019) http://dx.doi.org/10.1016/j.jksuci.2019.02.003.

[61] A.R. Wani, Q.P. Rana, U. Saxena, N. Pandey, Analysis and detection of DDoS attacks on cloud computing environment using machine learning techniques, in: Amity International Conference on Artificial Intelligence, IEEE, 2019, pp. 870–875, http://dx.doi.org/10.1109/AICAI.2019.8701238.

[62] M. Panda, A. Abraham, M.R. Patra, Discriminative multinomial naive bayes for network intrusion detection, in: International Conference on Information Assurance and Security, IEEE, 2010, pp. 5–10, http://dx.doi.org/10.1109/ISIAS.2010.5604193.

[63] S.K. Ajagekar, V. Jadhav, Study on web DDOS attacks detection using multinomial classifier, in: International Conference on Computational Intelligence and Computing Research, IEEE, 2016, pp. 1–5, http://dx.doi.org/10.1109/ICCIC.2016.7919656.

[64] S.K. Ajagekar, V. Jadhav, Automated approach for DDOS attacks detection based on naive Bayes multinomial classifier, in: International Conference on Trends in Electronics and Informatics, IEEE, 2018, pp. 1–5, http://dx.doi.org/10.1109/ICOEI.2018.8553848.

[65] H. Chen, M. Liu, F. Zhongchuan, Using improved Hilbert–Huang transformation method to detect routing-layer reduce of quality attack in wireless sensor network, Wirel. Pers. Commun. 104 (2) (2019) 595–615, http://dx.doi.org/10.1007/s11277-018-6036-3.

[66] A. Kuzmanovic, E.W. Knightly, Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants, in: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2003, pp. 75–86, http://dx.doi.org/10.1145/863955.863966.

[67] A. Shevtekar, N. Ansari, A router-based technique to mitigate reduction of quality (RoQ) attacks, Comput. Netw. 52 (5) (2008) 957–970, http://dx.doi.org/10.1016/j.comnet.2007.11.015.

[68] I. Sreeram, V.P.K. Vuppala, HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm, Appl. Comput. Inform. 15 (1) (2017) 59–66, http://dx.doi.org/10.1016/j.aci.2017.10.003.

[69] S.S. Mohammed, R. Hussain, O. Senko, B. Bimaganbetov, J. Lee, F. Hussain, C.A. Kerrache, E. Barka, M.Z.A. Bhuiyan, A new machine learning-based collaborative DDoS mitigation mechanism in software-defined network, in: 14th International Conference on Wireless and Mobile Computing, Networking and Communications, IEEE, 2018, pp. 1–8, http://dx.doi.org/10.1109/WiMOB.2018.8589104.

[70] A.M. Alrehan, F.A. Alhaidari, Machine learning techniques to detect DDoS attacks on VANET system: A survey, in: 2nd International Conference on Computer Applications & Information Security, IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/CAIS.2019.8769454.

[71] J. Hou, P. Fu, Z. Cao, A. Xu, Machine learning based DDos detection through netflow analysis, in: IEEE Military Communications Conference, IEEE, 2018, pp. 1–6, http://dx.doi.org/10.1109/MILCOM.2018.8599738.

[72] C. Balarengadurai, S. Saraswathi, Detection of exhaustion attacks over IEEE 802.15.4 MAC layer using fuzzy logic system, in: 12th International Conference on Intelligent Systems Design and Applications, IEEE, 2012, pp. 527–532, http://dx.doi.org/10.1109/ISDA.2012.6416593.

[73] J.C.C. Rodriguez, A.P. Briones, J.A. Nolazco, FLF4DoS. Dynamic DDoS mitigation based on TTL field using fuzzy logic, in: 17th International Conference on Electronics, Communications and Computers, IEEE, 2007, p. 12, http://dx.doi.org/10.1109/CONIELECOMP.2007.19.

[74] H.S. Mondal, M.T. Hasan, M.B. Hossain, M.E. Rahaman, R. Hasan, Enhancing secure cloud computing environment by detecting DDoS attack using fuzzy logic, in: 3rd International Conference on Electrical Information and Communication Technology, IEEE, 2017, pp. 1–4, http://dx.doi.org/10.1109/EICT.2017.8275211.

[75] A. Alsirhani, S. Sampalli, P. Bodorik, DDoS detection system: Using a set of classification algorithms controlled by fuzzy logic system in apache spark, IEEE Trans. Netw. Serv. Manag. 16 (3) (2019) 936–949, http://dx.doi.org/10.1109/TNSM.2019.2929425.

[76] M.H. Bhuyan, E. Elmroth, Multi-scale low-rate DDoS attack detection using the generalized total variation metric, in: 17th IEEE International Conference on Machine Learning and Applications, IEEE, 2018, pp. 1040–1047, http://dx.doi.org/10.1109/ICMLA.2018.00170.

[77] W. Ren, D.-Y. Yeung, H. Jin, M. Yang, Pulsing RoQ DDoS attack and defense scheme in mobile ad hoc networks, Int. J. Netw. Secur. 4 (2) (2007) 227–234.

[78] M. Guirguis, A. Bestavros, I. Matta, Y. Zhang, Reduction of quality (RoQ) attacks on dynamic load balancers: Vulnerability assessment and design tradeoffs, in: 26th International Conference on Computer Communications, IEEE, 2007, pp. 857–865, http://dx.doi.org/10.1109/INFCOM.2007.105.

[79] M. Guirguis, A. Bestavros, I. Matta, Y. Zhang, Adversarial exploits of end-systems adaptation dynamics, J. Parallel Distrib. Comput. 67 (3) (2007) 318–335, http://dx.doi.org/10.1016/j.jpdc.2006.10.005.
[80] Y. Chen, K. Hwang, Spectral analysis of TCP flows for defense against reduction-of-quality attacks, in: International Conference on Communications, IEEE, 2007, pp. 1203–1210, http://dx.doi.org/10.1109/ICC.2007.204.
[81] W. Chen, Y. Zhang, Y. Wei, The feasibility of launching reduction of quality (RoQ) attacks in 802.11 wireless networks, in: 14th International Conference on Parallel and Distributed Systems, IEEE, 2008, pp. 517–524, http://dx.doi.org/10.1109/ICPADS.2008.59.
[82] S.A. Arunmozhi, Y. Venkataramani, A flow monitoring scheme to defend reduction-of-quality (RoQ) attacks in mobile ad-hoc networks, Inf. Secur. J. Glob. Perspect. 19 (5) (2010) 263–272, http://dx.doi.org/10.1080/19393555.2010.514651.
[83] S. Gulati, A.S. Dhaliwal, Mitigating RoQ attacks using flow monitoring method, Int. J. Eng. Trends Technol. 4 (2013) 4074–4079.
[84] A. Jain, D. Zongker, Feature selection: Evaluation, application, and small sample performance, IEEE Trans. Pattern Anal. Mach. Intell. 19 (2) (1997) 153–158, http://dx.doi.org/10.1109/34.574797.
[85] Y. Zhou, G. Cheng, S. Jiang, M. Dai, Building an efficient intrusion detection system based on feature selection and ensemble classifier, Comput. Netw. (2020) 17, http://dx.doi.org/10.1016/j.comnet.2020.107247.
[86] C. Khammassi, S. Krichen, A NSGA2-LR wrapper approach for feature selection in network intrusion detection, Comput. Netw. (2020) 18, http://dx.doi.org/10.1016/j.comnet.2020.107183.
[87] M. Wang, Y. Lu, J. Qin, A dynamic MLP-based DDoS attack detection method using feature selection and feedback, Comput. Secur. 88 (2020) 1–14, http://dx.doi.org/10.1016/j.cose.2019.101645.
[88] S.M.T. Nezhad, M. Nazari, E.A. Gharavol, A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks, IEEE Commun. Lett. 20 (4) (2016) 700–703, http://dx.doi.org/10.1109/LCOMM.2016.2517622.
[89] K.S. Sahoo, D. Puthal, M. Tiwary, J.J.P.C. Rodrigues, B. Sahoo, R. Dash, An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics, Future Gener. Comput. Syst. 89 (2018) 685–697, http://dx.doi.org/10.1016/j.future.2018.07.017.
[90] K. Kumar, R.C. Joshi, K. Singh, A distributed approach using entropy to detect DDoS attacks in ISP domain, in: International Conference on Signal Processing, Communications and Networking, IEEE, 2007, pp. 331–337, http://dx.doi.org/10.1109/ICSCN.2007.350758.
[91] R. Wang, Z. Jia, L. Ju, An entropy-based distributed DDoS detection mechanism in software-defined networking, in: Trustcom/BigDataSE/ISPA, volume 1, IEEE, 2015, pp. 310–317, http://dx.doi.org/10.1109/Trustcom.2015.389.
[92] A.T. Lawniczak, H. Wu, B. Di Stefano, Entropy based detection of DDoS attacks in packet switching network models, in: International Conference on Complex Sciences, Springer, 2009, pp. 1810–1822, http://dx.doi.org/10.1007/978-3-642-02469-6_57.
[93] S.S. Kolahi, A.A. Alghalbi, A.F. Alotaibi, S.S. Ahmed, D. Lad, Performance comparison of defense mechanisms against TCP SYN flood DDoS attack, in: 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, IEEE, 2014, pp. 143–147, http://dx.doi.org/10.1109/ICUMT.2014.7002093.
[94] J.V.P. Gomes, Classification of Peer-to-Peer traffic by exploring the heterogeneity of traffic features through entropy, Universidade da Beira Interior, 2012.
[95] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (3) (1948) 379–423, http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x.
[96] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[97] E. Sanchez, T. Shibata, L.A. Zadeh, Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives, World Scientific, 1997.
[98] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, Int. J. Hum.-Comput. Stud. 51 (2) (1999) 135–147, http://dx.doi.org/10.1016/S0020-7373(75)80002-2.
[99] caida.edu, Center for applied internet data analysis, 2016, URL: https://data.caida.org/datasets/security/ddos-20070804/$ddostrace.20070804_144936.pcap.gz$. (Accessed 27 November 2016).
[100] Y. Xiang, K. Li, W. Zhou, Low-rate DDoS attacks detection and traceback by using new information metrics, IEEE Trans. Inf. Forensics Secur. 6 (2) (2011) 426–437, http://dx.doi.org/10.1109/TIFS.2011.2107320.
[101] Netkit, Netkit, 2016, URL: http://wiki.netkit.org/. (Accessed 15 November 2016).
[102] Curl-loader, Curl-loader, 2017, URL: http://curl-loader.sourceforge.net. (Accessed 09 June 2017).
[103] Wireshark, Wireshark, 2018, URL: https://www.wireshark.org/docs/man-pages/tshark.html. (Accessed 09 April 2018).
[104] M-RoQ, Manipulated-RoQ attack software, 2020, URL: https://github.com/ducarios/M-RoQ. (Accessed 28 February 2020).
[105] T.T.T. Nguyen, G.J. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutor. 10 (1–4) (2008) 56–76, http://dx.doi.org/10.1109/SURV.2008.080406.
[106] psrecord, Record the CPU and memory activity of a process, 2019, URL: https://github.com/astrofrog/psrecord. (Accessed 19 June 2019).

Vinícius de Miranda Rios was born in Janaúba, Minas Gerais, Brazil, in 1980. Holds a 4-year B.Sc. degree in Information Systems, obtained from the Centro Universitário Luterano do Brasil (CEULP-ULBRA), Brazil, in 2005 and 2-year Master degree in Electrical Engineering, obtained from the Universidade de Brasília (UnB), Brazil, in 2012. Actually he is a Ph.D. student at the Universidade da Beira Interior (UBI), Covilhã, Portugal, through a Ph.D. grant from the Brazilian CAPES foundation. He is a professor of Computer Science at Instituto Federal de Educação, Ciência e Tecnologia do Tocantins (IFTO) since 2015, where he lectures subjects related with computer network and programming to graduate courses.

Pedro R.M. Inácio was born in Covilhã, Portugal, in 1988. He Holds a 5-year B.Sc. degree in Mathematics/Computer Science and a Ph.D. degree in Computer Science and Engineering, obtained from the Universidade da Beira Interior (UBI), Portugal, in 2005 and 2009 respectively. The Ph.D. work was performed in the enterprise environment of Nokia Siemens Networks Portugal S.A., through a Ph.D. grant from the Portuguese Foundation for Science and Technology. He is a professor of Computer Science at UBI since 2010, where he lectures subjects related with information assurance and security, programming of mobile devices and computer based simulation, to graduate and undergraduate courses, namely to the B.Sc., M.Sc. and Ph.D. programmes in Computer Science and Engineering. He is currently the Head of the Department of Computer Science of UBI. He is an instructor of the UBI Cisco Academy. He is an IEEE senior member and a researcher of the Instituto de Telecomunicações (IT). His main research topics are information assurance and security, computer based simulation, and network traffic monitoring, analysis and classification. He has about 30 publications in the form of book chapters and papers in international peer-reviewed books, conferences and journals. He frequently reviews papers for IEEE, Springer, Wiley and Elsevier journals. He has been member of the Technical Program Committee of international conferences such as the ACM Symposium on Applied Computing - Track on Networking.

Damien Magoni is a full professor of computer science at the University of Bordeaux since 2008. From 2002 to 2008, he was an associate professor at the University of Strasbourg. His main research interests are in computer communications and networking, with a focus on Internet architecture, protocols, and applications. Some of his research has been supported by grants from the European Union, the CNRS, and Science Foundation Ireland. He has co-published over 80 refereed research papers. He also has authored several open-source software for networking research and teaching. His latest contributions are the virtual network device and the network mobilizer, which jointly enable the emulation of mobile networks. He earned a MEng in 1995 from Télécom Paris, as well as a MSc in 1999 and a Ph.D. in 2002 both from the University of Strasbourg. He has been a visiting researcher at various institutions around the world, including the AIST at Tsukuba, the University of Sydney, the University of Michigan at Ann Arbor, and University College Dublin. He has reviewed for over 20 academic journals and has been in the TPC of numerous high-level conferences. He is a senior member of both the IEEE and the ACM.

Mário M. Freire received the five-year BS degree in Electrical Engineering and the two-year MS degree in Systems and Automation in 1992 and 1994, respectively, from the University of Coimbra, Portugal. He received the Ph.D. degree in Electrical Engineering in 2000 and the Habilitation title in Computer Science in 2007 from the University of Beira Interior (UBI), Portugal. He is a full professor of Computer Science at UBI, which he joined in the Fall of 1994. In April 1993, he did one-month internship at the Research Centre of Alcatel-SEL (now Nokia Networks) in Stuttgart, Germany. His main research interests fall within the area of computer systems and networks, including network and systems virtualization, cloud and edge computing and security and privacy in computer systems and networks. He is the co-author of seven international patents, co-editor of eight books published in the Springer LNCS book series, and co-author of about 130 papers in international journals and conferences. He serves as a member of the editorial board of the ACM SIGAPP Applied Computing Review, serves as associate editor of the Wiley Security and Privacy journal and of the Wiley International Journal of Communication Systems, and served as editor of IEEE Communications Surveys and Tutorials in 2007–2011. He served as a technical program committee member for several IEEE international conferences and is co-chair of the track on Networking of ACM SAC 2020. Dr. Mário Freire is a chartered engineer by the Portuguese Order of Engineers and he is a member of the IEEE Computer Society and of the Association for Computing Machinery.