Computer Networks 235 (2023) 109951

Contents lists available at ScienceDirect

Computer Networks

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

Analyzing the influence of the sampling rate in the detection of malicious traffic on flow data

Adrián Campazas-Vega [a],*, Ignacio Samuel Crespo-Martínez [b], Ángel Manuel Guerrero-Higueras [a], Claudia Álvarez-Aparicio [a], Vicente Matellán [a], Camino Fernández-Llamas [a]

[a] Robotics Group, University of León, Campus de Vegazana s/n, 24071 León, Spain
[b] Supercomputación Castilla y León (SCAYLE), Campus de Vegazana s/n, 24071 León, Spain

ARTICLE INFO

Keywords:
Network security
Sampling flow data
Malicious traffic detection

ABSTRACT

Cyberattacks are a growing concern for companies and public administrations. The literature shows that analyzing network-layer traffic can detect intrusion attempts. However, such detection usually implies studying every datagram in a computer network. Therefore, routers routing a significant volume of network traffic do not perform an in-depth analysis of every packet. Instead, they analyze traffic patterns based on network flows. However, even gathering and analyzing flow data has a high-computational cost, and therefore routers usually apply a sampling rate to generate flow data. Adjusting the sampling rate is a tricky problem. If the sampling rate is low, much information is lost and some cyberattacks may be neglected, but if the sampling rate is high, routers cannot deal with it. This paper tries to characterize the influence of this parameter in different detection methods based on machine learning. To do so, we trained and tested malicious-traffic detection models using synthetic flow data gathered with several sampling rates. Then, we double-check the above models with flow data from the public BoT-IoT dataset and with actual flow data collected on RedCAYLE, the Castilla y León regional academic network.

1. Introduction

Detecting malicious traffic and analyzing complete network packets is a problem currently addressed using different methods. Studying packets' payload is ideal but unrealistic scenario in core routers since studying packets' payload in real-time is highly CPU-demanding and the volume of traffic in these routers is huge. Therefore, these routers use flow data-based analysis instead of packet-based to detect network anomalies.

A flow is defined as a set of IP packets passing an observation point in the network that share a set of common properties such as source and destination IP address, and source and destination port number [1]. Network flow data generated by routers does not store packets payload, only features relevant to the whole flow.

There are different flow network protocols such as sFlow, CFlowd and NetStream. One of the most widespread is NetFlow, designed by Cisco Inc. to collect, aggregate and record traffic flow data in their routers [2]. NetFlow collects features such as the number of packets in the flow, the IP protocol type, the protocol flags, and the time stamp. NetFlow does not store packet payloads, losing most information in network communication but reducing the computational cost of analyzing it. NetFlow has been instrumental in the development of the Internet Protocol Flow Information Export (IPFIX) [3], protocol by the Internet Engineering Task Force (IETF) [4]. IPFIX serves as an established industry standard for exporting comprehensive network flow information. This protocol enables the systematic collection of data pertaining to network traffic in a structured manner, facilitating its transmission to a centralized collector or analyzer for in-depth analysis. IPFIX is considered an evolution of NetFlow.

It has also been shown in the literature that it is possible to detect malicious traffic on flow data using machine learning. For instance, authors in [5] successfully detected application-layer Distributed Denial of Service (DDoS) attacks on NetFlow data – specifically Slow Read attacks – using up to six classification algorithms. Also, in [6], different types of malicious traffic are detected in other flow-based datasets using a decision tree-based model. We have also proven [7] that it is possible to detect SQL Injection attacks on NetFlow flows collected without packet sampling.

However, some routers manage such a significant volume of network traffic that even gathering flow data is too CPU-demanding [8].

* Corresponding author.
E-mail addresses: acamv@unileon.es (A. Campazas-Vega), icrem@unileon.es (I.S. Crespo-Martínez), am.guerrero@unileon.es (Á.M. Guerrero-Higueras), calvaa@unileon.es (C. Álvarez-Aparicio), vmato@unileon.es (V. Matellán), camino.fernandez@unileon.es (C. Fernández-Llamas).

https://doi.org/10.1016/j.comnet.2023.109951
Received 12 April 2023; Received in revised form 29 June 2023; Accepted 25 July 2023
Available online 28 July 2023

These routers sample the traffic when generating flow information to avoid overloading. In this context, "sampling rate" refers to the number of sequential data packets pass before one sample is collected to generate the flow information.

Router manufacturers recommend low sampling rates. For instance, Juniper recommends sampling rates lower than one of every one thousand packets (1/1000).[1] But it is up to the network managers to tune this parameter. Determining the compromise between the detection of malicious traffic capability versus the load of the routers, is one of the goals of the work described in this paper.

1.1. Research motivation

This research arises from the need to detect malicious traffic in networks that use flow-based protocols and packet sampling specifically in networks using NetFlow V5 protocol to mitigate the computational burden on the routers. We focus on networks that use high sampling rates. Unfortunately, there are very few proposals for this problem. In particular, the primary motivation of this work was to provide network managers with a rationale of how sampling rate value could influence the ability to detect cyberattacks and which machine learning techniques could work better for detecting the attacks depending on the sampling rate used. Additionally, it aims to provide evidence of these studies by providing public datasets and tools that could be used to replicate the results or improve them.

This paper also intends to demonstrate that it is possible to detect attacks in sampled network flow data using machine-learning techniques. Finally, this work shows that the DOROTHEA tool [9] – Our tool developed to generate datasets based on network flows – can generate NetFlow datasets with different sampling thresholds that can be used to train algorithms for their use on real traffic.

In order to validate the proposals described, some of our experiments have been carried out using real network traffic from the regional academic network of Castilla *y* León (RedCAYLE), part of the national Spanish academic and research network (RedIRIS [10]), which provides advanced communication services to the scientific community and to the Spanish universities affiliated to the network. RedCAYLE serves nine universities, several university hospitals, and more than 1360 schools across one of the largest regions in Europe. RedCAYLE traffic is managed mainly by two routers in two different cities, plus a set of auxiliary routers distributed throughout the region. Finally, the BoT-IoT dataset[2] has been used to validate our models against a well-known dataset.

In summary, this paper presents three main contributions:

1. We show that it is possible to detect malicious traffic in network infrastructures that gather flow data with a sampling threshold of 1 out of 1000 packets. We do not believe this has been demonstrated in the literature so far.
2. We determine the best algorithm for malicious traffic detection depending on the sampling rate used to collect flow data.
3. We empirically demonstrate that systems fitted with datasets generated by DOROTHEA could be successfully transferred to detect malicious traffic in real scenarios.

Additionally, this work includes the collection and publication of seven datasets comprising packet-sampled flow data. These datasets are made available under a free-use license, allowing other researchers to utilize them for verifying the results and claims presented in this paper.

The remainder of the paper is organized as follows. Section 2 describes the current state of the art in malicious traffic detection. Section 3 describes the tools, the experiments carried out, and the methodology used to evaluate our proposal. Section 4 shows the results obtained in the above experiments. The results are discussed in Section 5. Finally, conclusions are posed in Section 6.

2. Related work

Numerous proposals exist in the literature to detect malicious traffic by analyzing network packets. Machine learning-based detection models specifically supervised learning-based models, are highly popular. In [11], the authors carry out a literature review map to establish the most common algorithms and datasets used to detect network attacks. For instance, in [12], the authors obtain a 94.36% accuracy score using an Averaged One-Dependence Estimator (AODE), a 92.70% accuracy score using a Bayesian Network-based model, and a 75.73% accuracy score using a Naive Bayes (NB)-based model. The above models were trained and tested with the UNSWNB15 dataset. Similar work was carried out in [13]. Here, the authors not only focused on the model's effectiveness but also on its efficiency. Using the same dataset, the authors conclude that AODE is the best algorithm, with a 97.26% accuracy score and a running time of about 7 s. Using a more complex approach, the authors in [14] propose a hybrid model using bagging and rotation forest techniques, obtaining an 85.8% accuracy score. In [15], the authors propose a graphical features-based method, getting a 98.54% accuracy score with a K-Nearest Neighbors (KNN) model. In [11], the authors conclude that the best algorithms for detecting network attacks are KNN, Decision Tree (DT), and NB.

However, there is not much research in the literature that attempts to detect malicious traffic in network flows collected by applying packet sampling. In [16], the performance of a DT-based model for detecting malicious traffic in packet-based data is studied on a NetFlow-based dataset. The results show that the adapted model obtains similar accuracy when using network packets and when dealing with flow-based data without packet sampling. However, accuracy highly decreases when applying a sampling rate. For example, for a sampling rate of 1/100, the authors achieve an overall accuracy of 85%. However, with a sampling threshold of 1 out of 1000 packets, the authors still need to obtain an accuracy greater than 50%.

Authors in [17] propose the Smart Detection system, an online approach for DoS/DDoS attack detection. This software uses the Random Forest (RF) algorithm to classify network traffic on samples obtained from network devices using the sFlow protocol.[3] Results show an online detection rate above 96% using a 20% sampling rate. The work in [18] explores the impact of packet sampling on the performance of machine learning-based network intrusion systems using three sampling rates – 1/10, 1/100, and 1/1000 –. In their experiments, the authors use a Convolutional Neural Network (CNN), DT, and RF algorithms on datasets gathering DoS and brute-force attacks. Results show that 50% of the malicious flows are not detected even with a 1/10 sampling rate. Authors in [19], use a CNN for port scan detection on sampled NetFlow Version 5 data. The authors generate a graphical representation of flow data to train and test the system. They obtain a 94.15% of accuracy with a sampling rate of 1/500. However, with a sampling rate of 1/1000, the accuracy decrease to 50%.

The work carry out in [20] analyze the flow-based CIDDS-001 dataset using KNN and k-means algorithms.

In [9], we present DOROTHEA. This tool allows deploying complex network architectures to subsequently collect network flows. we used DOROTHEA to generate 2 datasets to detect port scanning attacks. The classification models used were trained with port scanning attacks and validated with a second dataset containing slow port scanning attacks. The results show that the network flows contain enough information to detect port scanning attacks.

[1] See \textit{TrafficSampling,Forwarding,andMonitoringOverview} on Juniper TechLibrary.

[2] https://research.unsw.edu.au/projects/bot-iot-dataset

[3] https://sflow.org/

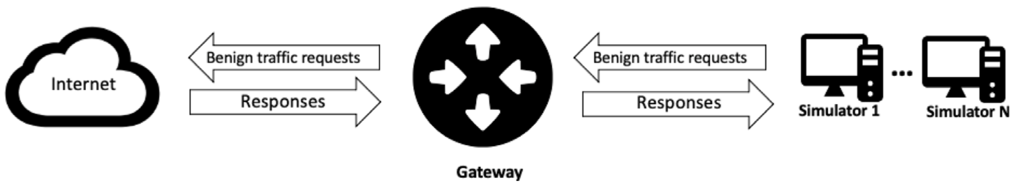


Fig. 1. Benign traffic generation scheme.

As mentioned above, detecting malicious traffic on flow data is possible without packet sampling. In addition, some works have obtained good results in detecting malicious traffic with a small packet sampling threshold. However, they fail when the sampling threshold increases to values similar to those used in real networks [16].

3. Tools and methods

This section presents the tools used and the experiments carried out. First in Section 3.1 we describe the technical specifications of the computer where the models were trained and tested. Secondly, we briefly describe DOROTHEA and the flow data collection process in Section 3.2. Next, we describe RedCAYLE's architecture and the flow-data gathering in a state-of-the-art environment in Section 3.3. Section 3.4 describes the BoT-IoT dataset and the flow data gathering from the network packets in the dataset. Data processing is analyzed in Section 3.5. Then, we describe Model Evaluator (MoEv), the tool we used to generate our machine learning-based detection models in Section 3.6. Finally, in Section 3.7, we present the evaluation methodology.

3.1. Experimental setup

The experiments carried out in this work were performed using a desktop computer. The system had 32 GB of RAM, a 3.60 GHz Intel Core i5 8600K processor with 6 cores, an Nvidia GeForce GTX 1050Ti graphics card, a 500 GB hard drive and was running the Ubuntu 20.04 operating system.

3.2. DOROTHEA

DOROTHEA [21] is a Docker-based framework for gathering NetFlow data. Thanks to the flexibility and scalability provided by Docker, DOROTHEA allows deploying different complex network architectures in order to generate and collect flows in NetFlow v5, v9 and IPFIX format. DOROTHEA deploys a NetFlow sensor, which generates flows from network packets passing through a given network interface. Furthermore, the generation of malicious traffic and the generation of benign traffic are executed on distinct, mutually isolated frameworks. Consequently, the tool enables unambiguous labeling of the traffic it generates.

The benign traffic generation framework simulate the usual network traffic that can be generated by a set of users in a company or a public organization. To do so, the tool uses several scripts written in Python that generate network traffic corresponding to web browsing, SSH connections, and sending emails. DOROTHEA is a modular tool that allows users to modify existing scripts or even incorporate new scripts. Network traffic is managed by a gateway that performs two main tasks. On the one hand, it routes the generated packets to the Internet. On the other hand, it sends 1 out of $\mathcal{X}$ packets – where $\mathcal{X}$ is the sampling threshold set by the user – to a NetFlow data generation node. Fig. 1 shows the benign traffic generation process.

The framework that generate malicious traffic are isolated from the Internet. This ensures that all generated network flows correspond to network attacks. The scheme of the malicious traffic generators is shown in Fig. 2. As with the benign traffic generators, attacks are launched using Python scripts, which can be customized by users. The gateway works as explained above by routing packets and collecting sampled flow data. DOROTHEA is available under an open license.[4]

Table 1
NetFlow V5 features [7].

Feature	Description
sysuptime	Current time in milliseconds since the export device started
unix_secs	Current count of seconds since 0000 UTC 1970
unix_nsecs	Residual nanoseconds since 0000 UTC 1970
engine_type	Flow switching motor type
engine_id	Slot number switching engine flow
exaddr	Flow exporter IP address
srcaddr	Source IP address
dstaddr	Destination IP address
nexthop	IP address of the next hop router
input	SNMP index of the input interface
output	SNMP index of the exit interface
dpkts	number of packets contained in the flow
dockets	Total number of bytes of layer 3 in the packets of the flow
first	Sysuptime at start of flow
last	Sysuptime when the last packet in the flow was received
srcport	TCP/UDP source port number
dstport	TCP/UDP destination port number
tcp_flags	TCP flags
prot	IP type of protocol (e.g., TCP = 6; UDP = 17)
tos	IP type of service (ToS)
src_as	Autonomous system number of the source
dst_as	Autonomous system number of the destination
src_mask	Source address prefix mask bits
dst_mask	Destination address prefix mask bits

Data collection with DOROTHEA. We created up to six flow datasets with DOROTHEA [22], three for training the models – $\mathcal{D}_1$–$\mathcal{D}_3$ – and the remaining for testing them – $\mathcal{D}_4$–$\mathcal{D}_6$ –. These datasets were collected by applying different packet sampling thresholds: 1/250; 1/500; and 1/1000.

Table 1 shows the features we collect in $\mathcal{D}_1$–$\mathcal{D}_6$ datasets before preprocessing. They correspond to the NetFlow version 5 fields. Although DOROTHEA works with several NetFlow versions, version 5 was chosen because that is the version RedCAYLE uses, as shown later in Section 3.3. Unlike other well-known datasets such as NSL-KDD and UNSW-NB15, which gather network packets, or CIDDS-001, which collects flow data without applying any sampling rate, our datasets contain flow data generated at different sampling rates.

$\mathcal{D}_1$–$\mathcal{D}_6$ includes both benign and malicious traffic. In our case, malicious traffic is generated by carrying out port scanning attacks. Both training and test sets collect flow data with different sampling thresholds: 1 out of 250 packets, 1 out of 500, and 1 out of 1000. All datasets gathered are balanced in terms of the amount of malicious traffic and benign traffic they contain. The percentage of malicious and benign traffic is around 50% to prevent classifiers from always predicting the majority class. Benign flow data is labeled “0”, and malicious flow data “1”.

In order to include malicious traffic on the training sets, port scanning attacks were carried out using Nmap by running the following TCP and UDP scan types: TCP SYN; TCP Connect; UDP; TCP NULL, FIN,

[4] https://github.com/uleroboticsgroup/DOROTHEA

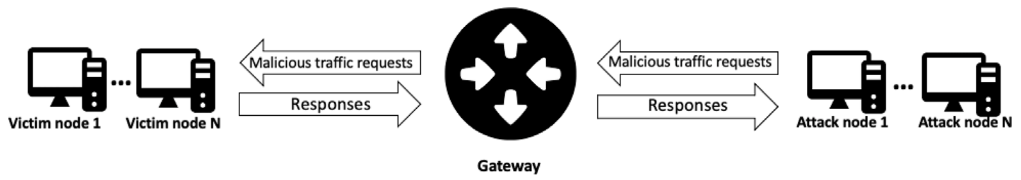


Fig. 2. Malicious traffic generation scheme.

and Xmas-tree; TCP ACK; TCP Window; and TCP Maimon scanning, see [23]. The attacks ran on 100 nodes that scanned 65,536 ports on 200 victim nodes. The IP address space was 182.168.1.1/24 for benign- and malicious-traffic-generation nodes. For the victim nodes, the address space was 126.52.30.0/24.

Regular port-scanning attacks were carried out for generating the training sets, and slow port-scanning attacks for the test sets. Specifically, port scan requests were launched with a 5- to 10-second slack time between them. The introduction of slack time between each packet results in a significant increase in the duration of the attack, primarily due to the substantial volume of packets being transmitted. Networks' IP address spaces were also different in the test and training sets. On test sets, the IP address space was 152.148.48.1/24 for benign and malicious traffic generation nodes and 140.30.20.1/24 for victim nodes.

DOROTHEA requires separating the benign-malicious traffic generation process for empirically labeling flows as benign or malicious, so traffic is not simultaneously gathered, which is one of its drawbacks since it impacts the nature of the sampling process. Thus, proposed models are also double-checked with other datasets not affected by this issue, as described in the following sections.

3.3. RedCAYLE

In order to evaluate the systems fitted with synthetic datasets produced by DOROTHEA, real NetFlow flows were obtained from the RedCAYLE network's routers. RedCAYLE provides their affiliated institutions – educational centres, University hospitals, scientific infrastructures, and technological facilities – with a high-capacity communications backbone network infrastructure, thus allowing access to research network resources and the Internet. In the educational community alone, the network supports more than 380,000 students and teachers from Castilla *y* León.

RedCAYLE provides a wide variety of services: 10 Gbps point-to-point transport service, Internet connection, IP addressing, incident management, and service monitoring. The monitoring service allows the affiliated institutions to analyze and diagnose the status of their network services. To do so, RedCAYLE uses NetFlow version 5 for its network analysis. The NetFlow implementation allows for a statistics-based approach, as exhaustively examining each packet within the network is infeasible due to computational limitations. In order to computationally analyze the traffic handled by the network, it is not sufficient to use network flows. It is imperative to employ a sampling rate. According to the guidelines provided by the manufacturer of RedCAYLE's router, a sampling threshold of 1 packet per 1000 is advised.

Data collection from RedCAYLE. To double-check our detection models fitted with DOROTHEA and to demonstrate that they work in a real environment, flow data from RedCAYLE [22] – $\mathcal{D}_7$ – is used. Since RedCAYLE uses NetFlow version 5, the features collected are the same depicted in Table 1. To generate the malicious traffic, we carried out new port scanning attacks against nodes within the network range of RedCAYLE. The attacks were made from a known IP address range, so all the flows that have an IP address from such range correspond were considered malicious traffic and labeled as "1". Flows corresponding to benign traffic, labeled as "0", were randomly collected from the flow data generated in RedCAYLE. Every flow collected in this dataset has been generated with a 1/1000 sampling rate. Regarding the algorithm used by the routers to select the packet that will become part of a flow, Juniper does not provide information about their sampling algorithms.

It is crucial to emphasize that although the dataset used in this work was collected on routers manufactured by Juniper, the collected data are in NetFlow v5 format. This format is a standardized protocol for flow data export, and it is implemented not only by Juniper routers but also by other manufacturers such as Cisco [24] and Enterasys Switches [25]. Therefore, the insights and findings derived from the analysis of this dataset collected on Juniper routers can be considered applicable and relevant to other network devices that utilize NetFlow v5.

3.4. BoT-IoT

In addition to the datasets collected with DOROTHEA, and the one gathered from RedCAYLE life traffic, the models have also been tested with flow data from the BoT-IoT dataset [26]. The BoT-IoT dataset include 470,655 packets specifically associated with port scans. This high number was the criteria used to choose this dataset because a large amount of information gets lost when sampling is applied. Therefore, a large number of packets is necessary to obtain meaningful results.

Data collection form BoT-IoT. This dataset comprises network packets. In order to get sampled flow data, it must be transformed. Network packets were sampled by selecting 1 out of 1000 packets, then flows have been generated using Softflowd [27]. The resulting dataset is named $\mathcal{D}_8$.

3.5. Data curation

Data curation has been carried out to improve the performance and to reduce the bias that the models may have due to the nature of the data. First, datasets are checked to eliminate flows that were not generated correctly – some feature of Table 1 was not collected correctly –. Next, we extract features with a variance score of 0. Specifically, features 'exaddr', 'engine_type', 'engine_id', 'src_mask', 'dst_mask', 'src_as', and 'dst_as' have been removed. Additionally, time-related features – 'unix secs', 'unix_nsecs', 'sysuptime', 'first', and 'last' – are also removed to remove the time bias. Finally, the features associated with specific IP addresses have also been removed to prevent the models from generating biases depending on the IP address in the training set. Specifically, 'srcaddr', 'dstaddr', and 'nexthop' features have been removed.

After data curation, the remaining features are 'dpkts', 'dockets', 'input', 'output', 'srcport', 'dstport', 'prot', 'tos', and 'tcp_flags'.

3.6. Model fitting

We used MoEv [28] to prepare our detection models. MoEv is a wrapper for the Scikit-learn [29] API that allows for automatically building classification models from labeled datasets. MoEv allows performing multiple tasks on the datasets it receives such as cleaning, dimensionality reduction or scaling. For model training, MoEv allows performing hyperparameter optimization using GridSearchCV.

In addition, MoEv uses the Dask library [30] to parallelize the requested tasks. MoEv generates a report with multiple indicators such as Accuracy, False Alarm Rate (FAR), Matthews correlation coefficient, Cohen's kappa coefficient, Precision, Recall, and F$_1$ score. It is used in different research areas such as jamming attack detection on real-time location systems [31] and students' academic success prediction [32]. Furthermore, in [7,9], MoEv has been successfully used to build malicious-traffic detection models on unsampled flow data.

As the objective of the models is to predict a category for each network flow – (0) for benign traffic and (1) for malicious traffic – a priori classification algorithms are more suitable for this task than algorithms based on regression or clustering. However, it has been shown that in complex problems more value is placed on the data than on the type of algorithm used [33,34]. Therefore, both classification and regression-based algorithms have been evaluated. MoEv was used to fit, tune hyperparameters – the hyperparameters obtained for each model are shown in Section 4 –, and test the detection models. Specifically, the following algorithms were computed: KNN [35], Logistic Regression (LR) [36], Linear Support Vector Classification (LSVC) [37], LSVC with Stochastic Gradient Descent (SGD) [38], Multi-layer Perceptron (MLP) [39] and, RF as a collection of decision trees [40].

3.7. Evaluation

To evaluate the experiment, several KPIs were calculated from the confusion matrix generated by each model. First, the accuracy score of the models was calculated as shown in Eq. (1), where T_P is the number of malicious samples correctly identified as malicious. T_N points to the number of harmless or benign samples correctly identified as benign traffic. F_P is the number of benign flows misclassified as malicious. Finally, F_N points to the number of malicious flows misclassified as benign traffic.

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \tag{1}$$

Furthermore, the following Key Performance Indicators (KPIs) have been considered: FAR, Matthews correlation coefficient (ϕ), Cohen's kappa coefficient (κ), Precision ($\mathcal{P}$), Recall ($\mathcal{R}$), and F$_1$ score ($\mathcal{F}_1$).

False Alarm Rate (FAR) represents the proportion of benign flows erroneously classified as malicious flows, as shown in Eq. (2).

$$\text{FAR} = \frac{F_P}{T_N + F_P} \tag{2}$$

ϕ is often used to measure the quality of binary classifiers. ϕ is a more reliable statistical rate which produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset [41], as shown in Eq. (3).

$$\phi = \frac{T_P \times T_N - F_P \times F_N}{\sqrt{(T_P + F_P)(T_P + F_N)(T_N + F_P)(T_N + F_N)}} \tag{3}$$

$\mathcal{P}$ measures the accuracy of the positive predictions, as shown in Eq. (4).

$$\mathcal{P} = \frac{T_P}{T_P + F_P} \tag{4}$$

$\mathcal{R}$, also called sensitivity or true positive rate, indicates the proportion of positive examples that are correctly identified by the model among all real positives, as shown in Eq. (5).

$$\mathcal{R} = \frac{T_P}{T_P + F_N} \tag{5}$$

It is often convenient to combine Precision and Recall into the F$_1$ score, particularly if a simple way to compare two classifiers is needed. $\mathcal{F}_1$ relates Recall and Precision, being the harmonic mean of both values. While the regular mean treats all values equally, the harmonic mean gives much more weight to low values. It computes as shown in Eq. (6).

$$\mathcal{F}_1 = 2\frac{\mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}} \tag{6}$$

Cohen's kappa coefficient (κ) [42] measures the agreement between two raters who classify N items into C mutually exclusive categories. To compute κ, we need the relative observed agreement among raters, and the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly seeing each category. If the raters are in complete agreement, then $\kappa = 1$. If there is no agreement among the raters other than what would be expected by chance, then $\kappa = 0$. The statistic can be negative if there is no relationship between the ratings of the two raters, or to reflect a real tendency of the raters to give differing ratings [43].

Since our models are binary classifiers – i.e. $C = 2$ –, we can compute κ as shown in Eq. (7).

$$\kappa = \frac{2 \times (TP \times TN - FN \times FP)}{(TP + FP) \times (FP + TN) + (TP + FN) \times (FN + TN)} \tag{7}$$

The results obtained in this study from the analysis of sampled network flows were statistically compared with the detection models described in [9], which were trained with unsampled flow data. In addition, a performance comparison between the model provided in [19], and our detection models was carried out.

Table 2
Accuracy, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ obtained in [9] for malicious-traffic detection models on not-sampled flow data.

Algorithm	Accuracy	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$
KNN	0.964	0.965	0.964	0.964
LR	0.948	0.951	0.948	0.948
LSVC	0.932	0.938	0.932	0.932
LSVC+SGD	0.921	0.929	0.921	0.921
RF	0.902	0.914	0.902	0.901

4. Results

Table 2 summarizes the results previously shown in [9] for malicious-traffic detection models on not-sampled flow data. Specifically, the table shows the Accuracy, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ for KNN-, LR-, LSVC-, LSVC+SGD-, and RF-based models. These results are compared with the ones obtained by the malicious-traffic detection models on sampled flow data proposed in this work. The MLP model has not been compared because it was not evaluated in [9].

The malicious-traffic detection models on sampled flow data were trained using the previously-described $\mathcal{D}_1$–$\mathcal{D}_3$ datasets. Then, models were tested with $\mathcal{D}_4$–$\mathcal{D}_8$. Table 3 summarize the aim, the source, the sampling rate, the number of samples, and the benign-malicious traffic ratio for each dataset. As explained in Section 3, $\mathcal{D}_1$ was used to train detection models on flow data with a sampling threshold of 1 out of 250 packets. $\mathcal{D}_4$ was used to test them. Next, $\mathcal{D}_2$ was used to train detection models on flow data with a sampling threshold of 1 out of 500 packets. $\mathcal{D}_5$ was used to test them. Finally, $\mathcal{D}_3$ was used to train detection models on flow data with a sampling threshold of 1 out of 1000 packets. $\mathcal{D}_6$, $\mathcal{D}_7$ and $\mathcal{D}_8$ was used to test them.

A Jupyter Notebook that allows for replicating the evaluation carried out in this work is available online in a Binder-ready repository.[5] All datasets used in this research are uploaded to the datasets folder within the repository and are also available on Zenodo [22].

After tuning, the following hyperparameters were selected for each model (visit the Jupyter Notebook for details):

[5] https://github.com/uleroboticsgroup/MoEv/tree/Sampling

Table 3
Dataset volumetry.

Dataset	Source	Aim	Sampling rate	Samples	Benign-malicious ratio
$\mathcal{D}_1$	DOROTHEA	Train	1/250	10,416	50%
$\mathcal{D}_2$	DOROTHEA	Train	1/500	2,996	50%
$\mathcal{D}_3$	DOROTHEA	Train	1/1,000	2,894	50%
$\mathcal{D}_4$	DOROTHEA	Test	1/250	18,292	50%
$\mathcal{D}_5$	DOROTHEA	Test	1/500	5,454	50%
$\mathcal{D}_6$	DOROTHEA	Test	1/1,000	2,646	50%
$\mathcal{D}_7$	RedCAYLE	Test	1/1,000	920	50%
$\mathcal{D}_8$	BoT-IoT	Test	1/1,000	7,653	50%

Table 4
Accuracy, FAR, ϕ, κ, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ of KNN-, LR-, LSVC-, LSVC+SGD-, MLP-, and RF-based detection models on sampled flow data collected in DOROTHEA.

Sampling rate: 1/250.							
Algorithm	Accuracy	FAR	ϕ	κ	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$
KNN	0.957	0.014	0.916	0.915	0.959	0.957	0.957
LR	0.945	0.000	0.896	0.891	0.951	0.946	0.946
LSVC	0.917	0.000	0.847	0.835	0.918	0.918	0.918
LSVC+SGD	0.998	0.000	0.997	0.997	0.998	0.998	0.998
MLP	0.968	0.025	0.937	0.930	0.968	0.968	0.968
RF	0.913	0.045	0.829	0.825	0.916	0.912	0.912

Sampling rate: 1/500.							
Algorithm	Accuracy	FAR	ϕ	κ	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$
KNN	0.955	0.003	0.914	0.911	0.959	0.955	0.955
LR	0.631	0.000	0.389	0.263	0.788	0.632	0.574
LSVC	0.661	0.404	0.438	0.323	0.797	0.662	0.618
LSVC+SGD	0.925	0.000	0.859	0.849	0.935	0.925	0.924
MLP	0.939	0.024	0.881	0.878	0.941	0.939	0.939
RF	0.922	0.013	0.852	0.845	0.929	0.922	0.922

Sampling rate: 1/1000.							
Algorithm	Accuracy	FAR	ϕ	κ	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$
KNN	0.971	0.000	0.943	0.941	0.974	0.974	0.974
LR	0.520	0.479	0.040	0.040	0.520	0.520	0.520
LSVC	0.755	0.328	0.000	0.000	0.836	0.755	0.739
LSVC+SGD	0.767	0.314	0.595	0.534	0.831	0.767	0.755
MLP	0.911	0.049	0.826	0.822	0.914	0.911	0.911
RF	0.921	0.028	0.847	0.842	0.926	0.921	0.920

- **KNN**. We used twenty-five neighbors for the neighbor queries. As a weight function, we used the Euclidean distance.
- **LR**. We used a regularized version of linear regression, specifically Ridge regression.
- **LSVC**. We used a linear kernel function. The regularization parameter (C) is set to 1.0. As a loss function, we used Hinge.
- **LSVC+SGD**. We used the same parameters as LSVC but applied stochastic gradient descent.
- **MLP**. We used a neural network with 10 hidden layers. The activation function of these layers is the logistic sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$
- **RF**. We trained with 20 trees in the forest. The minimum number of samples required to split an internal node is 2, and the minimum number of samples required to be at a leaf node is 1.

Fig. 3 shows the confusion matrices of malicious-traffic detection models obtained after testing on sampled flow data. According to the confusion matrices in the figure, Table 4 shows Accuracy, FAR, ϕ, κ, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ of KNN-, LR-, LSVC-, LSVC+SGD-, MLP-, and RF-based malicious-traffic detection models on sampled flow data.

In order to double-check our malicious-traffic detection models and demonstrate that they work in real environments, we also tested our models on flow data from RedCAYLE – $\mathcal{D}_7$ – and BoT-IoT dataset – $\mathcal{D}_8$ –. Fig. 4 shows the confusion matrices obtained. According to the above confusion matrices, Tables 5 and 6 show the Accuracy, FAR, ϕ, κ, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ of our detection models after testing on $\mathcal{D}_7$ and $\mathcal{D}_8$.

Finally, the performance of the models has been evaluated in terms of execution time or computational efficiency. The time taken by the models to predict a single sample was evaluated. Therefore, the performance of the models remains linear regardless of the number of flows processed in a production environment. Section 3.1 shows the configuration of the machine on which the experiment was performed. This evaluation provides insight into the practical feasibility of the proposed models for real-time deployment in network security systems. Table 7 presents the execution time per sample, measured in microseconds μs, for the models utilized in this study.

5. Discussion

One issue that can be observed in Table 3 is that as the sampling rate decreases, the number of flows of the datasets also decreases. This is due to the fact that as the sampling rate decreases more packets are required to generate a flow.

Focusing on the accuracy scores shown in Table 2, we can see that all the malicious-traffic detection models computed achieve an accuracy higher than 90% on not-sampled flow data. Table 4 shows the performance obtained by detection models on sampled flow data. All malicious-traffic detection models achieve an accuracy higher than 90% on flow data with a 1/250 packet sampling rate. For a packet sampling rate of 1/500, the LSVC and LR models drastically reduce their detection capability with an accuracy less than 64%. The remaining four models – KNN, LSVC+SGD, MLP, and RF – keep their detection capability with an accuracy higher than 90%. Using a sampling rate of 1/1000, LSVC+SGD also loses its detection capability. Only three models keep an accuracy higher than 90%. KNN is the best model with an accuracy score of 97%.

According to the above, we can claim that if the routers computationally could afford a sampling rate of 1/250, all the algorithms used in our experiments would have a good detection capability and thus improve network security. As the sampling rate decreases, some models lose their detection capability, e.g. LR and LSVC with a 1/500 sampling rate. As shown in Table 4, just KNN-, MLP-, and RF-based models keep an accuracy score above 90% with a 1/1000 sampling rate. Therefore, using a 1/1000 sampling rate, as most commercial routers do, makes KNN-, decision tree-, or deep learning-based models the most suitable algorithms to detect malicious traffic on sampled flow data.

It is important to point out that although some models keep similar accuracy with extremely high sampling thresholds, it does not mean that the network's security will be the same. For instance, with a 1/1000 sampling rate, a much larger amount of information is lost than using a sampling of 1/500 or 1/250. Therefore, the probability that a malicious flow is gathered and later detected is lower. Thus, decreasing the sampling rate will always decrease the network's security.

Table 5
Accuracy, FAR, ϕ, κ, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ of KNN-, LR-, LSVC-, LSVC+SGD-, MLP-, and RF-based detection models trained with the dataset $\mathcal{D}_3$ – collected in DOROTHEA–, and testing on $\mathcal{D}_7$ – collected in RedCAYLE–. Both datasets with a sampling rate of 1/1000.

Algorithm	Accuracy	FAR	ϕ	κ	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$
KNN	0.971	0.036	0.944	0.943	0.972	0.972	0.972
LR	0.560	0.386	0.137	0.122	0.577	0.560	0.536
LSVC	0.656	0.407	0.431	0.313	0.796	0.656	0.610
LSVC+SGD	0.643	0.416	0.409	0.287	0.792	0.643	0.592
MLP	0.922	0.076	0.843	0.843	0.922	0.922	0.922
RF	0.908	0.154	0.831	0.817	0.923	0.908	0.907

Table 6
Accuracy, FAR, ϕ, κ, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ of KNN-, LR-, LSVC-, LSVC+SGD-, MLP-, and RF-based detection models trained with the dataset $\mathcal{D}_3$ – collected in DOROTHEA–, and testing on $\mathcal{D}_8$ – BoT-IoT dataset–. Both datasets with a sampling rate of 1/1000.

Algorithm	Accuracy	FAR	ϕ	κ	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}_1$
KNN	0.834	0.004	0.707	0.668	0.874	0.834	0.829
LR	0.584	0.422	0.169	0.169	0.585	0.584	0.584
LSVC	0.470	0.532	−0.059	−0.059	0.470	0.470	0.470
LSVC+SGD	0.637	0.388	0.281	0.274	0.644	0.637	0.632
MLP	0.836	0.045	0.696	0.672	0.860	0.835	0.832
RF	0.835	0.043	0.695	0.670	0.860	0.836	0.833

Table 7
Execution time measured in microseconds of the models performing predictions per sample on the validation datasets $\mathcal{D}_4$–$\mathcal{D}_8$.

Algorithm	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	$\mathcal{D}_7$	$\mathcal{D}_8$
KNN	45.113 μs	34.610 μs	34.263 μs	30.642 μs	39.158 μs
LR	0.923 μs	0.828 μs	0.843 μs	1.485 μs	1.038 μs
LSVC	0.319 μs	0.302 μs	0.598 μs	1.284 μs	0.343 μs
LSVC+SGD	0.142 μs	0.290 μs	0.696 μs	2.218 μs	0.298 μs
MLP	3.863 μs	0.976 μs	1.023 μs	1.178 μs	0.891 μs
RF	0.375 μs	0.906 μs	0.783 μs	2.020 μs	0.474 μs

Regarding FAR, it behaves the same as the accuracy score. It tends to approach 0 when the models get a high accuracy score and to increase when their accuracy score decreases. Therefore, the best FAR score is obtained by the KNN-based model, followed by the RF- and MLP-based models. ϕ, κ, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ work inversely to FAR, increasing when the accuracy score increases and decreasing otherwise. Fig. 5 intuitively illustrates such a tendency with all the sampling thresholds studied. The figure also includes the performance of the models trained with not-sampled flow data proposed at [9] and summarized in Table 2.

As depicted above, our malicious-traffic detection models on sampled flow data were trained using the previously-described $\mathcal{D}_1$–$\mathcal{D}_3$ datasets. Then, models were tested with $\mathcal{D}_4$–$\mathcal{D}_6$. Next, to double-check them, they were also tested on $\mathcal{D}_7$, the dataset collected in RedCAYLE. As shown in Table 5 the KNN-, MLP-, and RF-based models generalize correctly, showing an accuracy score of 97.1%, 92.2%, and 90.8%, respectively. As with $\mathcal{D}_6$, the LSVC-, LSVC+SGD-, and LR-based models do not detect malicious traffic in flow data with a 1/1000 sampling rate. The remaining KPIs follow the same trend. The above results demonstrate that the KNN, MLP and RF models trained with the datasets collected in DOROTHEA allow detecting malicious traffic in sampled flow data in real environments, improving network security.

In addition to testing with RedCAYLE's flow data, we also evaluated the generalization capability of our malicious-traffic detection models with flow data – $\mathcal{D}_8$ – obtained from an external dataset, BoT-IoT. Table 6, shows the Accuracy, FAR, ϕ, κ, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{F}_1$ scores. The algorithms that get the best performance in $\mathcal{D}_8$ are, one more time, KNN, MLP, and RF. However, even obtaining promising results, all the KPIs were worse by between 10% and 20% concerning the performance obtained in the testing with RedCAYLE's flow data. This loss of performance is due to two fundamental factors. On the one hand, the packet sampling has not been performed in a router or using a NetFlow sensor – see Section 3.4 –. On the other hand, if network packets in the original BoT-IoT dataset are analyzed in-depth, we see that it is not a completely clean dataset. While it is true that the vast majority of the packets labeled as port scanning attacks correspond to port scans, a small portion contains other sorts of network traffic, such as DNS requests/responses.

In relation to the performance of the models in terms of execution times – see Table 7 –, out of the three models that exhibit satisfactory results in malicious traffic detection – KNN, MLP, and RF –, the KNN model demonstrates the poorest performance, exceeding 30 μs for all test datasets. Conversely, both the MLP and RF models showcase significantly improved execution times compared to the KNN model. The MLP model achieves execution times of less than 3.9 μs, while the RF model achieves execution times below 2 μs for all test datasets. The disparity in execution times between the KNN model and the other two models can be attributed to the inherent limitations of the KNN algorithm in terms of scalability. The KNN model requires access to the entire training dataset during classification, which becomes problematic when dealing with large datasets. Furthermore, the model computes the distance between the K nearest neighbors, which leads to increased runtime in proportion to the dataset's size. In fact, when considering deployment in production environments, the MLP and RF models may be more suitable choices compared to the KNN model, despite the KNN model exhibiting slightly higher detection capabilities. The limitations of the KNN model in terms of scalability and computational efficiency make it less practical for real-time deployment in network security systems.

Finally, as mentioned above in Section 1, no previous research in the literature aims to detect malicious traffic in 1/1000 sampled flow data. The best-performing proposal is [19], where the authors built images from flow data to train a ResNet-18 model. Table 8 illustrates the comparison between the ResNet-based model and our detection models. As shown in the table, all models get similar results with a 1/500

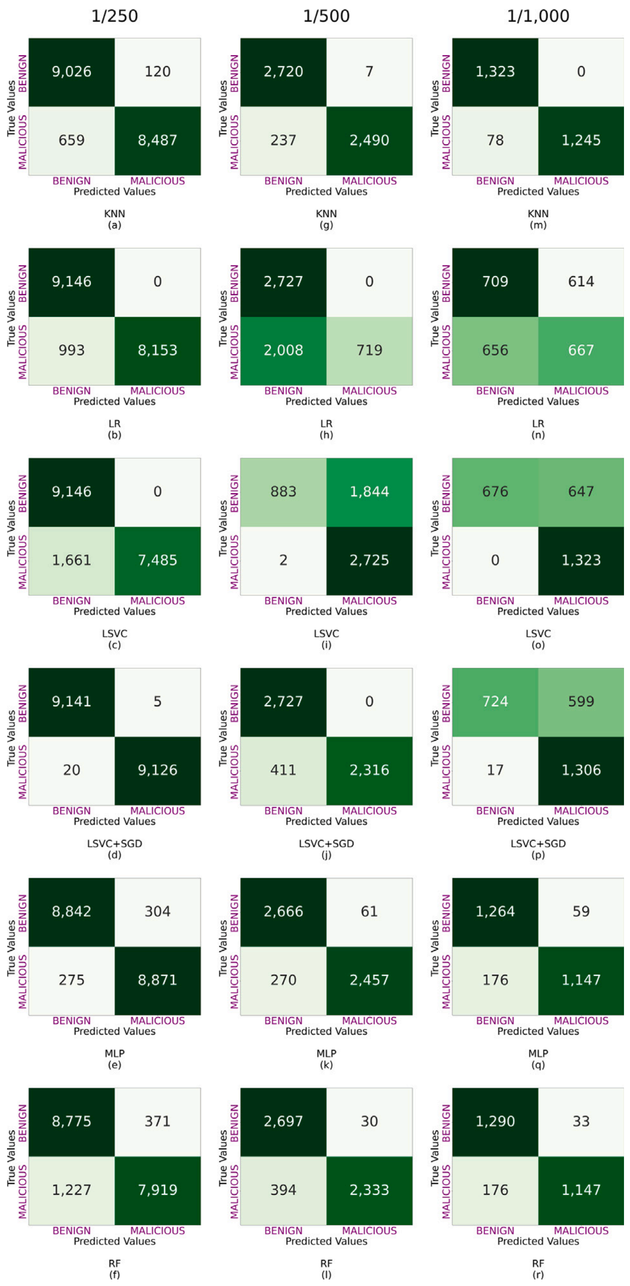


Fig. 3. Confusion matrices of KNN-, LR-, LSVC-, LSVC+SGD-, MLP-, and RF-based detection models on sampled flow data: 1/250 (a–f), 1/500 (g–l), and 1/1000 (m–r).

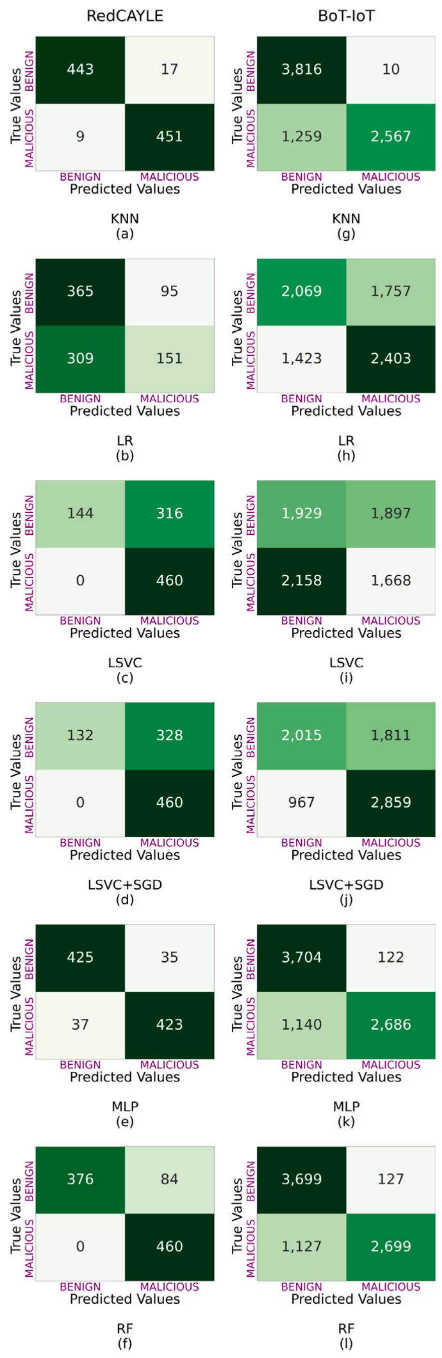


Fig. 4. Confusion matrices of KNN-, LR-, LSVC-, LSVC+SGD-, MLP-, and RF-based detection models tested on D_7 (a–f) and D_8 (g–l).

Table 8
Comparison of KNN, MLP, and RF Accuracy score vs. ResNet-18 [19].

Sampling rate	KNN	MLP	RF	ResNet-18 [19]
1/500	0.955	0.939	0.922	0.941
1/1,000	0.971	0.912	0.921	0.501

sampling rate. However, with a 1/1000 sampling rate, the ResNet-based model gets a 50% accuracy score versus the 97% accuracy score of our KNN-based model, the 91% accuracy score of our MLP-based model, and the 92% accuracy score of our RF-based model. This difference may be due to the technique used in [19] to build the images from flow data, i.e. due to the different features in our proposal. Finally, authors in [19] did not double-check their proposal, as we do with RedCAYLE's and BoT-IoT's flow data.

6. Conclusions

Machine learning-based models are commonly used to detect malicious network traffic by analyzing packet payloads. However, payload analysis is not feasible on wide-area networks where the amount of network traffic is significant. Such networks often use lightweight flow-based protocols like NetFlow to gather network traffic statistics. Flow data can also be used to detect malicious traffic, but collecting flow data is CPU demanding, so a sampling is usually applied. The research presented in this paper shows that machine learning-based models can also be used to detect malicious traffic in sampled flow data.

To demonstrate this, we collected flow datasets with different sampling thresholds using DOROTHEA: 1 out of 250, 1 out of 500, and

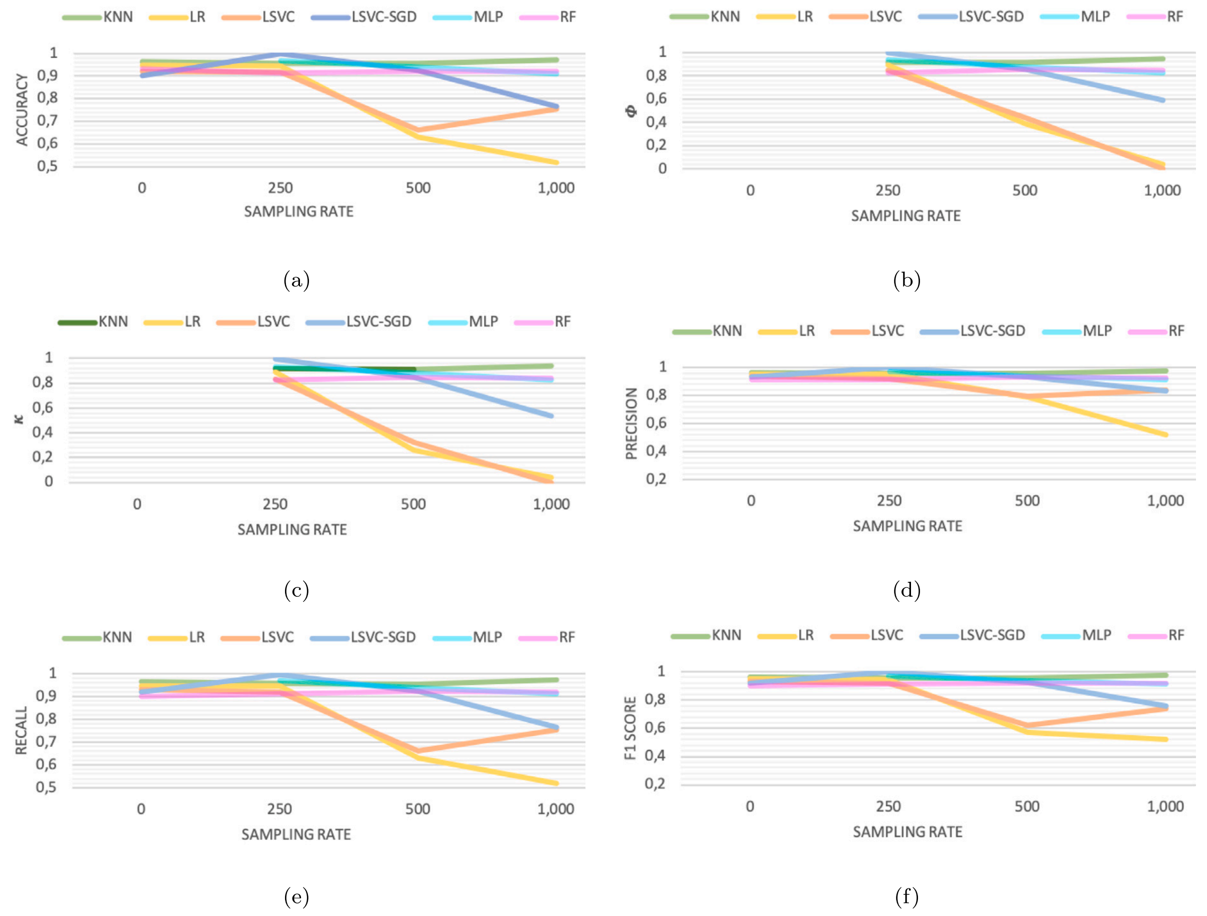


Fig. 5. Accuracy (a), ϕ (b), κ (c), $\mathcal{P}$ (d), $\mathcal{R}$ (e), and $\mathcal{F}_1$ (f) comparison of malicious-traffic detection models tested on sampled and not-sampled flow data.

1 out of 1000 packets. Machine learning-based models were trained and tested with the above datasets – specifically, KNN-, LR-, LSVC-, LSVC+SGD-, MLP-, and RF-based models –. Besides, to check that our malicious-traffic detection models generalize correctly in a realistic environment, we double-checked them on flow data collected from RedCAYLE, and with flow data from the well-known dataset, BoT-IoT.

We can conclude from the above experiments that malicious traffic may be detected in sampled flow data using machine learning-based detection models. However, the results change significantly depending on the sampling rate. With a sampling threshold of 1 out of 250 packets, all the algorithms tested achieved high detection rates. For a sampling rate of 1/500 packets, only the KNN, LSVC+SGD, MLP, and RF kept high detection rates. Finally, with 1/1000, the algorithms that show promising results are KNN, RF, and MLP. Specifically, the KNN-based one is the best model in terms of detection capability at all thresholds, but also the worst in execution performance. This consolidates the MLP model and RF model as more balanced options for deployment. It can also be concluded that the KNN-, MLP-, and RF-based models keep their detection capability even if the sampling rate significantly decreases. Our model's generalization was double-checked on actual flow data from RedCAYLE and the BoT-IoT dataset, empirically demonstrating that our detection models have good generalizability.

The aforementioned conclusions provide valuable insights into the performance of machine learning algorithms for detecting malicious traffic on sampled flow data, considering varying sampling rates. This knowledge is crucial for enhancing the security of important societal institutions like hospitals or universities.

DOROTHEA has been empirically proven to be a valuable tool for gathering flow datasets. Therefore, as future work, we intend to create new datasets collecting sampling flow data from further network attacks such as brute force, Denial of Service, or SQL Injection and also from other protocols such as IPFIX. we plan to use it to generate new detection models that improve the security of wide-area networks.

Acronyms

AODE	Averaged One-Dependence Estimator
CNN	Convolutional Neural Network
DOROTHEA	DOcker-based fRamework fOr gaTHering nEtflow dAta
DDoS	Distributed Denial of Service
DT	Decision Tree
FAR	False Alarm Rate
KPI	Key Performance Indicator
KNN	K-Nearest Neighbors
LR	Logistic Regression
LSVC	Linear Support Vector Classification
MLP	Multi-layer Perceptron
MoEv	Model Evaluator
RedCAYLE	Red de Ciencia y Tecnología de Castilla y León
RedIRIS	Red Académica y de Investigación Española
RF	Random Forest
SCAYLE	Supercomputación Castilla y León
SGD	Stochastic Gradient Descent
WAN	Wide Area Networks

CRediT authorship contribution statement

Adrián Campazas-Vega: Conceptualization, Software, Validation, Investigation, Data curation, Writing – original draft, Visualization.

Ignacio Samuel Crespo-Martínez: Software, Investigation, Data curation, Writing – review & editing. **Ángel Manuel Guerrero-Higueras:** Conceptualization, Methodology, Validation, Investigation, Writing – original draft, Visualization, Project administration. **Claudia Álvarez-Aparicio:** Software, Investigation, Writing – review & editing. **Vicente Matellán:** Conceptualization, Methodology, Resources, Writing – original draft, Supervision, Project administration. **Camino Fernández-Llamas:** Conceptualization, Methodology, Resources, 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.

Data availability

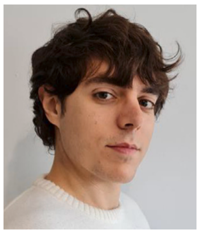
Data and tools are available in a public repository and on the Zenodo platform.

Acknowledgments

This research has been partially supported under the grant "Detección de nuevas amenazas *y* patrones desconocidos en la Red Regional de Ciencia *y* Tecnología", addenda 4 and 8 to the framework agreement between Instituto Nacional de Ciberseguridad de España (INCIBE) and Universidad de León 2019–2022 funded by INCIBE, Spain; and under the grant PID2021-126592OB-C21 funded by MCIN/AEI/10.13039/501100011033, Spain and by the "European Union".

References

[1] B. Claise, B. Trammell, P. Aitken, Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information, RFC 7011 (Internet Standard), Internet Engineering Task Force, 2013, pp. 2070–1721.
[2] J. Dreijer, NetFlow anomaly detection; finding covert channels on the network, 2014.
[3] IETF, IP flow information export (IPFIX) implementation guidelines, 2023, (Accessed Jun 8, 2023), https://datatracker.ietf.org/doc/html/rfc5153.
[4] IETF, Internet engineering task force, 2023, (Accessed Jun 8, 2023), https://www.ietf.org/.
[5] C. Kemp, C. Calvert, T. Khoshgoftaar, Utilizing netflow data to detect slow read attacks, in: 2018 IEEE International Conference on Information Reuse and Integration (IRI), IEEE, 2018, pp. 108–116.
[6] M. Sarhan, S. Layeghy, N. Moustafa, M. Portmann, Netflow datasets for machine learning-based network intrusion detection systems, in: Big Data Technologies and Applications, Springer, 2020, pp. 117–135.
[7] I.S. Crespo-Martínez, A. Campazas-Vega, Á.M. Guerrero-Higueras, V. Riego-DelCastillo, C. Álvarez-Aparicio, C. Fernández-Llamas, SQL injection attack detection in network flow data, Comput. Secur. 127 (2023) 103093.
[8] J.L. García-Dorado, J.E. López de Vergara, J. Aracil, V. López Álvarez, J.A. Hernández, S. López-Buedo, L.d. Pedro, Utilidad de los flujos NetFlow de RedIRIS para análisis de una red académica, in: RedIRIS: boletín de la Red Nacional de I+ D RedIRIS, RedIRIS, 2008.
[9] A. Campazas-Vega, I.S. Crespo-Martínez, Á.M. Guerrero-Higueras, C. Fernández-Llamas, Flow-data gathering using NetFlow sensors for fitting malicious-traffic detection models, Sensors 20 (24) (2020) 7294.
[10] RedIRIRS, About RedIRIRS, 2020, Online; accessed 11 November 2020, https://www.rediris.es/rediris/index.html.es.
[11] D. Sobrín-Hidalgo, A.C. Vega, Á.M.G. Higueras, F.J.R. Lera, C. Fernández-Llamas, Systematic mapping of detection techniques for advanced persistent threats, in: Conference on Complex, Intelligent, and Software Intensive Systems, Springer, 2020, pp. 426–435.
[12] M. Nawir, A. Amir, O.B. Lynn, N. Yaakob, R.B. Ahmad, Performances of machine learning algorithms for binary classification of network anomaly detection system, J. Phys.: Conf. Series 1018 (2018) 012015.
[13] M. Nawir, A. Amir, N. Yaakob, O.B. Lynn, Effective and efficient network anomaly detection system using machine learning algorithm, Bull. Electr. Eng. Inform. 8 (1) (2019) 46–51.
[14] B.A. Tama, M. Comuzzi, K.-H. Rhee, TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system, IEEE Access 7 (2019) 94497–94507, http://dx.doi.org/10.1109/ACCESS.2019.2928048.
[15] S. Chen, Z. Zuo, Z.P. Huang, X.J. Guo, A graphical feature generation approach for intrusion detection, in: W.-P. Sung, J. Kao (Eds.), MATEC Web of Conferences 44 (2016) 02041, http://dx.doi.org/10.1051/matecconf/20164402041.
[16] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, J. Solé-Pareta, Analysis of the impact of sampling on NetFlow traffic classification, Comput. Netw. 55 (5) (2011) 1083–1099.
[17] F.S.d. Lima Filho, F.A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar, L.F. Silveira, Smart detection: an online approach for DoS/DDoS attack detection using machine learning, Secur. Commun. Netw. 2019 (2019).
[18] J. Alikhanov, R. Jang, M. Abuhamad, D. Mohaisen, D. Nyang, Y. Noh, Investigating the effect of traffic sampling on machine learning-based network intrusion detection approaches, IEEE Access 10 (2021) 5801–5823.
[19] A.F.d. Retana, A. Miranda-García, Á.M. Guerrero, C. Fernández-Llamas, Attacks detection on sampled netflow traffic through image analysis with convolutional neural networks (CNN), in: Computational Intelligence in Security for Information Systems Conference, Springer, 2021, pp. 33–40.
[20] A. Verma, V. Ranga, Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning, Procedia Comput. Sci. 125 (2018) 709–716.
[21] A. Campazas-Vega, I.S. Crespo-Martínez, Á.M. Guerrero-Higueras, DOcker-based fRamework fOr gaTHering nEtflow dAta (DOROTHEA), Tech. rep., Robotics group, Universidad de León, 2020, http://dx.doi.org/10.5281/zenodo.4114119.
[22] A. Campazas-Vega, I.S. Crespo-Martínez, NetFlow data collected with different packet sampling rates (D1-D7), 2022, http://dx.doi.org/10.5281/zenodo.6243335, Online; accessed Feb 24, 2022.
[23] G.F. Lyon, Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning, Insecure, 2009.
[24] Cisco Systems, Inc, Cisco, 2022, (Accessed July 26, 2022), https://www.cisco.com/.
[25] Extreme Networks, Extreme networks, 2022, (Accessed Jun 10, 2023), https://www.extremenetworks.com/.
[26] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: BoT-IoT dataset, Future Gener. Comput. Syst. 100 (2019) 779–796.
[27] Damien Miller, Softflowd, 2022, (Accessed Jun 12, 2023), https://github.com/irino/softflowd.
[28] Á.M. Guerrero-Higueras, A. Campazas-Vega, I.S. Crespo-Martínez, Module evaluator (MoEv), Tech. rep., Robotics group, Universidad de León, 2020, http://dx.doi.org/10.5281/zenodo.4114127.
[29] 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.
[30] M. Rocklin, Dask: Parallel computation with blocked algorithms and task scheduling, in: Proceedings of the 14th Python in Science Conference, Vol. 130, Citeseer, 2015, p. 136.
[31] Á.M. Guerrero-Higueras, N. DeCastro-García, V. Matellán, Detection of cyber-attacks to indoor real time localization systems for autonomous robots, Robot. Auton. Syst. 99 (2018) 75–83.
[32] Á.M. Guerrero-Higueras, C. Fernández Llamas, L. Sánchez González, A. Gutierrez Fernández, G. Esteban Costales, M.Á.C. González, Academic success assessment through version control systems, Appl. Sci. 10 (4) (2020) 1492.
[33] M. Banko, E. Brill, Scaling to very very large corpora for natural language disambiguation, in: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, 2001, pp. 26–33.
[34] A. Halevy, P. Norvig, F. Pereira, The unreasonable effectiveness of data, IEEE Intell. Syst. 24 (2) (2009) 8–12.
[35] H. Mitchell, P. Schaefer, A "soft" K-nearest neighbor voting scheme, Int. J. Intell. Syst. 16 (4) (2001) 459–468.
[36] R.E. Wright, Logistic regression, 1995.
[37] C. Cortes, V. Vapnik, Support vector machine, Mach. Learn. 20 (3) (1995) 273–297.
[38] L. Bottou, Stochastic gradient learning in neural networks, Proc. Neuro-Nımes 91 (8) (1991) 12.
[39] G.E. Hinton, Connectionist learning procedures, in: Machine Learning, Elsevier, 1990, pp. 555–610.
[40] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[41] D. Chicco, G. Jurman, The advantages of the Matthews Correlation Coefficient (MCC) over F1 score and accuracy in binary classification evaluation, BMC Genomics 21 (2020) 1–13.
[42] J. Cohen, A coefficient of agreement for nominal scales, Educ. Psychol. Meas. 20 (1) (1960) 37–46.
[43] J. Sim, C.C. Wright, The kappa statistic in reliability studies: use, interpretation, and sample size requirements, Phys. Therapy 85 (3) (2005) 257–268.



Adrián Campazas Vega got his Ph.D. in Computer Science from the University of León in 2023. He got his degree in computer science (2015) and his master of science (2019) in Cybersecurity research at the University of León (León, Spain).

From 2015 to 2019 he worked in different companies as an application developer and cybersecurity auditor. In 2020, he became part of a cybersecurity research project for the University of León. In addition, from the academic year 2019 to 2023 he has been lecturer in the area of computer architecture at the University of León.

Ignacio S. Crespo Martinez got his degree in computer science (2017) and his master of science (2019) in Cybersecurity research at University of León (León, Spain). He is currently pursuing the Ph.D. in computer science at the University of León.

From 2018 to 2020 he has worked as a cybersecurity auditor. In 2020, he became part of a cybersecurity research project for the University of Leon.

Ángel Manuel Guerrero Higueras got his degree (2007) and his master of science (2010) in computer science at Rey Juan Carlos University (Madrid, Spain). Besides, he got his Ph.D. at the University of León in 2017.

He worked as an IT engineer at several companies in the private sector from 2000 to 2010 and from 2014 to 2016. In academia, he worked as a research assistant in the Atmospheric Physics Group (2011–2013) and in the Research Institute of Applied Science to CyberSecurity (2016–2018), both depending on the University of León. He currently stands as an Assistant Professor at the University of León. His main research interests include cybersecurity, robotics and AI

Claudia Álvarez Aparicio got her Ph.D. in Computer Science from the University of León in 2022.

From 2017 to 2020, she has been a Research Associate with the Research Institute of Applied Science to Cyber-Security and the Robotics Group at the University of León. From July 2020 to September 2021, she had a fellowship provided by the Regional Government of Castilla y León (Spain) to work on her Ph.D. Now she is Lecturer and researcher at the University of León. Her research interests include computer security and social robotics.

Vicente Matellán obtained his Ph.D in Computer Science from the Technical University of Madrid in 1993.

He was Assistant Professor at the University Carlos III of Madrid (Spain) (1993–1999) and Associate Professor at the University Rey Juan Carlos (Spain) from 1999–2008. In 2008 he joined the University of León (Spain), where he still serves as Full Professor in the Mechanical, Computer and Aerospace Engineering Department. His main research interests have to do with robotics, artificial intelligence, and cybersecurity where he has made more than 250 contributions in journals, books, and conferences.

Camino Fernández-Llamas got her Computer Science Engineering degree (1994), a MSc. on Knowledge and Software Engineering (1995) and her Ph.D. on Computer Science (2000) by the Polytechnic University of Madrid, Spain.

She started her teaching and researching career at University Carlos III of Madrid in 1995, and since 2008 she works for the University of León (Spain). Part of the Robotics Research Group, her main research interests are artificial intelligence and more specifically, human–machine and human–robot interaction, haptic simulation and cybersecurity. She has co-authored around 200 works in journals, conferences and workshops, and has been part or led more than 30 projects and contracts on these subjects.