A New Method for Flow-Based Network Intrusion Detection Using the Inverse Potts Model

Camila F. T. Pontes, Manuela M. C. de Souza, João J. C. Gondim, Matt Bishop, and Marcelo Antonio Marotta

Abstract—**Network Intrusion Detection Systems (NIDS) play an important role as tools for identifying potential network threats. In the context of ever-increasing traffic volume on computer networks, flow-based NIDS arise as good solutions for real-time traffic classification. In recent years, different flow-based classifiers have been proposed using Machine Learning (ML) algorithms. Nevertheless, classical ML-based classifiers have some limitations. For instance, they require large amounts of labeled data for training, which might be difficult to obtain. Additionally, most ML-based classifiers are not capable of domain adaptation, i.e., after being trained on an specific data distribution, they are not general enough to be applied to other related data distributions. And, finally, many of the models inferred by these algorithms are black boxes, which do not provide explainable results. To overcome these limitations, we propose a new algorithm, called Energy-based Flow Classifier (EFC). This anomaly-based classifier uses inverse statistics to infer a statistical model based on labeled benign examples. We show that EFC is capable of accurately performing binary flow classification and is more adaptable to different data distributions than classical ML-based classifiers. Given the positive results obtained on three different datasets (CIDDS-001, CICIDS17 and CICDDoS19), we consider EFC to be a promising algorithm to perform robust flow-based traffic classification.**

Index Terms—**Flow-based network intrusion detection, anomaly-based network intrusion detection, network flow classification, network intrusion detection systems, energy-based flow classifier, inverse Potts model, domain adaptation.**

I. INTRODUCTION

SYMANTEC'S Internet Security Threat Report [1] points out a 56% increase in the number of Web attacks in 2019. Network scans, denial of service, and brute force attacks are among the most common threats. Such malicious activities threaten individuals and collective organizations such as public health, financial, and government institutions. In this context, Network Intrusion Detection Systems (NIDSs) play an important role as tools for managing and identifying potential threats in the network [2].

There are two main approaches for NIDSs regarding the kind of data analyzed: packet-based and flow-based. In the former, deep packet inspection is performed, taking into account individual packet payloads and header information [3]. In the latter, flows, i.e., packet collections, are analyzed regarding their properties, e.g., duration, number of packets, number of bytes, and source/destination port [3]. To perform classification in real-time, a massive volume of data must be analyzed, making deep packet inspection too costly to be applied regarding processing and energy consumption. Since flow-based approaches can classify the whole traffic inspecting an equivalent to 0.1% of the total volume, NIDSs based on flow analysis arise as good solutions for real-time traffic classification [4]. Besides, with the advent of software-defined networking and the virtualization of network functions, distributed security systems can take advantage of the spread of NIDSs based on flow analysis to improve their security management across the network [5].

In recent years, different flow-based classifiers have been proposed based on both shallow and deep learning [6]. According to the report in [6], the best flow-based classifiers achieve around 99% accuracy. Although quite accurate, classical Machine Learning (ML)-based classifiers require labeled malicious traffic samples to perform training. However, real traffic labeling might be difficult, especially in the case of malicious traffic [7]. Besides, ML-based classifiers after trained on specific data distribution usually do not work well when applied to other data with slightly different distributions, i.e., they have low domain adaptation capability [8], [9], [10]. Such a capability is particularly important to the network context since a standard procedure is to perform the training of classifiers with simulated data and, afterward, apply in real scenarios that change the data distribution requiring domain adaptation. Moreover, most ML algorithms are well-known to be black-box mechanisms, challenging to be understood and readjusted in detail [11], [12]. In this regard, there is a clear need for a new flow-based classifier for NIDSs, which generates an understandable model (white box) based solely on benign examples, and adaptable to different domains.

In this work, we propose a novel classifier called Energy-based Flow Classifier (EFC), which is inspired by the inverse

Potts model from quantum mechanics and adapted to network flow classification. EFC performs one-class, anomaly-based classification, i.e., as long as it can learn the properties of benign flows, it will discriminate between benign and malicious flows. Moreover, it is a white box algorithm, producing a statistical model that can be analyzed in detail regarding individual parameter values. Here, we compared the performance of EFC against a variety of classifiers using three different datasets, i.e., CIDDS-001 [13], CICIDS17 [14], and CICDDoS19 [15]. Our results show that EFC's performance is comparable to the performance of the other classifiers. Also, we observed that EFC is less sensitive to changes in data distribution than the others. Our main contributions are:

- The proposal and implementation of a flow classifier based on the inverse Potts model to be employed in NIDSs;
- A performance comparison of the proposed classifier with classical ML-based classifiers using three different datasets;
- An analysis of how different classifiers perform when trained within one domain and tested in another related domain.

The rest of this paper is structured as follows. In Section II, we briefly present the state-of-the-art in flow-based NIDSs. In Section III, we describe the structure of network flows with a preliminary analysis of the datasets considered here. In Section IV, we introduce the statistical model proposed and the classifier implementation. In Section V, we present the results obtained regarding the statistical model's analysis and the classification experiments performed. Finally, in Section VI, we present our conclusions and future work.

II. RELATED WORK

In this section, we first briefly review the state-of-the-art in flow-based network intrusion detection systems. In the following, some previous work on CIDDS-001, CICIDS17, and CICDDoS19 are shown to highlight their relevance as up-to-date datasets to be used in our experiments. Finally, some challenges of ML-based traffic classification are regarded, such as the difficulty in obtaining sufficient labeled data, the non-interpretability of models, and the difficulty in adapting to different domains (data distributions).

Several ML-based classifiers have been explored over the last years for network intrusion detection. Vinayakumar *et al.* (2017) [16], Mahfouz *et al.* (2020) [17] and Khan *et al.* (2020) [18] independently evaluated the performance of different ML-based classifiers over Internet traffic datasets. In [16], the KDDCup'99 and NSL-KDD datasets are regarded to evaluate the performance of both shallow and deep learning-based classifiers. It is shown that deep learning-based approaches performed better to differentiate malicious attacks from benign traffic. Meanwhile, the authors of [17] considered the NSL-KDD dataset to compare the performance of different shallow learning-based classifiers. The classifier that presented the best performance without feature selection was the Decision Tree (DT). With feature selection, K-Nearest Neighbors (KNN) performed

better to classify malicious traffic. Finally, the authors of [18] compared the performance of a few different classifiers over the UNSW-NB15 dataset. They observed that Random Forest (RF) overperformed all other classifiers. In fact, RF has been used in several recent NIDSs [19], [20], [21]. All of the aforementioned works use the F1-score as a metric to assess the performance of the different classifiers. In the present work, we consider both deep and shallow learning-based classifiers as baselines to assess EFC's performance over three different datasets, regarding the F1-score as one of the evaluation metrics.

To assess EFC's performance, one of the datasets we use is CIDDS-001. This dataset was used by Verma and Ranga (2018) [22] to assess the performance of KNN and k-means clustering algorithms. Both algorithms achieved over 99% accuracy. Also, Ring *et al.* [23] explored slow port scans detection using CIDDS-001. The approach proposed by them is capable of accurately recognizing the attacks with a low false alarm rate. Finally, Abdulhammed *et al.* [24] also performed classification based on flows on CIDDS-001 and proposed an approach that is robust considering imbalanced network traffic. In summary, CIDDS-001 is an updated and relevant dataset to be used for network flow-classification solutions, being one of our dataset choices for assessing the performance of EFC.

The other two datasets considered here are CICIDS17 and CICDDoS19, from the Canadian Institute for Cyber Security. Recently, Yulianto *et al.* [25] used CICIDS17 to assess the performance of an Adaboost-based classifier. Aksu *et al.* [26] did the same in 2018 with different ML classifiers. CICIDS17 contains benign as well as the most up-to-date common attacks, resembling true real-world data, being a relevant dataset to consider for flow-based traffic classification. Meanwhile, CICDDoS19 is a very recent dataset with a focus on DDoS attacks. Reference [27] proposes a real-time entropy-based NIDS for detection of volumetric DDoS in Internet of Things (IoT) and performs tests over CICDDoS19 dataset, among other datasets. Another recent work [28] obtained over 99% accuracy over CICDDoS19 dataset using a Convolutional Neural Network (CNN). And, finally, Novaes *et al.* [29] proposed a system for intrusion detection based on fuzzy logic, which had its performance assessed on CICDDoS19. The rising popularity of this dataset serves as evidence of its relevance to assess the performance of different NIDS. Hence, we use CICIDS17 and CICDDoS19 datasets to test our classifier and compare it to the performance of classical ML classifiers.

Umer *et al.* (2017) [6] performed a comprehensive literature survey on flow-based network intrusion detection. In their work, some disadvantages of using ML-based classifiers for traffic classification are mentioned. Among them are the high computational cost of training the classifiers, the difficulty in obtaining representative datasets, and the high false positive rates observed. The present work addresses some of the issues mentioned, since the classifier proposed here has a low computational cost and learns exclusively based on benign samples. In the following, these and some other issues are discussed in further detail.

One of the commonly discussed issues in the field of ML is the tight dependency most algorithms have on the amount

of labeled samples available for training [7], which might be difficult to obtain in some contexts. For instance, it is difficult to obtain malicious traffic samples and to label it in the real world and this is why most of the network intrusion detection datasets contain simulated attacks. This issue makes it difficult to train intrusion detection algorithms in such a way that they might be able to detect zero-day threats [7]. The only way of possibly detecting a zero-day attack is relying on an anomaly-based classifier [30], such as the one we propose in this work. EFC, has a great advantage over other ML-based algorithms, which is the capability to infer a model based solely on benign traffic samples, i.e., half of the information. Such capability can be used to circumvent the problem of obtaining a high amount of data and the labeling of malicious samples.

Another common problem in ML is that inferred models lose their predictive performance when tested in different domains (data distributions) [10]. In the field of network security, this adaptability is specially important given the existence of zero-day threats and the artificiality of most datasets used for research. In [10], there is an interesting discussion about the existing differences between datasets used by academics to test NIDSs and the network traffic observed in the real world. Additionally, the work of Bartos *et al.* [8] and Li *et al.* [9] also address this issue. They propose similar approaches, applying data transformations to the data to reduce differences between data distributions in different domains. In our work, we propose a classifier that is intrinsically adaptable to different domains, since the model inference is based solely on benign samples. Therefore, there is no need to transform the data in order to adapt the model or perform adjustments to a different domain, making our approach simpler and more straightforward.

Finally, another big issue in ML is the non-interpretability of some models [11], [12]. Artificial Neural Networks (ANNs), in special, became more and more opaque with time, despite overperforming other approaches in many tasks. The authors of [12] highlight that the best ML algorithms are not interpretable, hence the decision taken by them can not be explained. However, different contexts require transparent decision making and that is why the development of explainable models is so important. The authors of [11] call attention to the fact that trying to explain black box models might not be the best approach to solve the issue of non-interpretability. Instead, it is suggested the design of new models that are inherently interpretable. In line with what has been suggested by these recent studies, EFC generates a white-box model and, therefore, satisfies the requirement of providing explainable results, allowing classification results to be analysed in retrospect if needed. Thus, next, we introduce main concepts and intuitions to serve as basis for EFC.

III. BACKGROUND AND DATASETS

In this section, we present some fundamental concepts to understand flow-based network intrusion detection. First, the concept of network flow and its features are introduced. In the following, the three Internet flow datasets used in this work are presented and described in detail to provide concrete

examples of features and contextualize the experimental results presented in Section V. The information provided in this section serves as a basis to understand how EFC works.

A network flow is a set of packets that traverses intermediary nodes between end-points within a given time interval. Under the perspective of an intermediary node, i.e., an observation point, all packets belonging to a given flow have a set of common features called flow keys. It means that flow keys do not change for packets belonging to the same flow, while the remaining features might vary. FlowScan [31] is an example of a tool capable of collecting data from a set of packets and extracting flow features to be later exported in different formats, such as NetFlow and IPFIX. Since NetFlow is the most commonly used format, its main features are listed below:

- Source/Destination IP (flow keys) - determine the origin and destination of a given flow in the network;
- Source/Destination port (flow keys) - characterize different kinds of network services, e.g., ssh service uses port 22;
- Protocol (flow key) - characterizes flows regarding the transport protocol used, e.g., TCP, UDP, ICMP.
- Number of packets (feature) - total number of packets captured in a flow;
- Number of bytes (feature) - total number of bytes in a flow;
- Duration (feature) - total duration of a flow in seconds;
- Initial timestamp (feature) - system time when one flow started to be captured.

Other features such as TCP Flags and Type of Service might also be exported in some cases. The combination of different flow keys and features characterize one flow and determine its particular behavior.

Flow-based approaches are seen as suitable alternatives to precede packet inspection in real-time NIDSs. The idea is to deeply inspect only the packets belonging to flows considered to be suspicious by the flow-based classifier. A two-step approach would notably reduce the amount of data analyzed while maintaining a high classification accuracy [4]. In this work, we are only concerned with the first step, which is the flow classification. We evaluate the performance of our algorithm, the EFC, compared to other ML algorithms using three different datasets. We also evaluate the performance of the algorithms by training with data from part of the dataset and testing with other parts of it. Nonetheless, although both parts of the data come from the same dataset, their distributions are different to characterize domain adaptation. In the following, we briefly describe the datasets used for testing and characterize what constitutes a domain adaptation in each of them.

A. CIDDS-001

CIDDS-001 [13] is a relatively recent dataset composed of a set of flow samples captured within a simulated OpenStack environment and another set of flow samples obtained from a real server. The former contains only simulated traffic, while the latter includes both real and simulated traffic. Each sample collected within these two environments has one of the labels described in Table I.

TABLE I
LABELS WITHIN CIDDS-001 DATASET

Environment	Labels
OpenStack	normal, DoS, portScan, pingScan, bruteForce
External server	normal, DoS, bruteForce, unknown, suspicious

TABLE II
FEATURES WITHIN CIDDS-001 DATASET

#	Name	Description
1	Src IP	Source IP Address
2	Src Port	Source Port
3	Dest IP	Destination IP Address
4	Dest Port	Destination Port
5	Proto	Transport Protocol (*e.g.,* ICMP, TCP, or UDP)
6	Date first seen	Start time flow first seen
7	Duration	Duration of the flow
8	Bytes	Number of transmitted bytes
9	Packets	Number of transmitted packets
10	Flags	OR concatenation of all TCP Flags

TABLE III
ATTACKS WITHIN CICIDS17 DATASET

Week day	Attacks
Monday	
Tuesday	FTP-Patator, SSH-Patator
Wednesday	DoS slowloris, DoS Slowhttptes, DoS Hulk, DoS GoldenEye, Heartbleed Port 444
Thursday	Brute Force, XSS, Sql Injection, Dropbox download, Cool disk
Friday	Botnet ARES, Port Scan, DDoS LOIT

TABLE IV
ATTACKS WITHIN CICDDoS19 DATASET

Day	Attacks
First	PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, SYN
Second	NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDos, SYN, TFTP

Simulated benign flows are labeled as *normal*, while simulated malicious flows are labeled as *dos*, *portScan*, *pingScan* or *bruteForce*, depending on the type of attack simulated. The labels *suspicious* and *unknown*, in turn, are used for real traffic. The external server is open to user access through the ports 80 and 443. Hence, flows directed at these ports were labeled as *unknown*, since they could be either benign or malicious. All flows directed at other ports were labeled as *suspicious*. Traffic was sampled in both the simulated and the external server environment for a period of four weeks. Within this dataset, a change from the simulated data distribution to the external server data distribution is a domain change, requiring the classifiers to adapt.

CIDDS-001 dataset flow features are shown in Table II. All features were taken into account for characterization and classification except for *Src IP*, *Dest IP* and *Date first seen*. These exceptions are because the latter one is intrinsically not informative to differentiate flows, and the former two are made up in the context of the simulated network and might be confounding.

B. CICIDS17

CICIDS17 [14] dataset contains benign traffic and the most up-to-date common attacks, resembling real-world data. This dataset was built using the abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and e-mail protocols. The data was captured during one week in July 2017. The attacks implemented include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. They were executed both morning and afternoon on Tuesday, Wednesday, Thursday, and Friday (see Table III).

Flow features on this dataset were extracted using CICFlowMeter [32]. There are in total 88 features, which are not going to be cited here because of the limited space. All features were considered here, except for Flow ID, Source IP, Destination IP, and Timestamp. These exceptions were made because the features were either intrinsically not informative or made up within a simulated environment.

C. CICDDoS19

CICDDoS19 [15] contains benign traffic and the most up-to-date common DDoS attacks (volumetric and application: low volume, slow rate), resembling real-world data. This dataset contains different modern reflective DDoS attacks such as PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, SYN, NTP, DNS, and SNMP. The traffic was captured in January (first day) and March (second day), 2019. Attacks were executed during this period (see Table IV).

Flow features on this dataset were extracted using CICFlowMeter [32]. All features were considered here, except for Flow Id, Source IP, Destination IP, and Timestamp. These exceptions were made because the features were either intrinsically not informative or made up within a simulated environment.

Considering each concept regarding network flows, their features, and how they are presented across different datasets, serve as basis to introduce the main intuition behind EFC, such as presented next.

IV. PROPOSAL

EFC is based on inverse statistics. The main task of inverse statistics is to infer a statistical distribution based on a sample of it [33]. Methods using inverse statistics have been successfully applied to problems in other disciplines, e.g., predicting protein contacts in Biophysics [33], [34]. Here, the statistical inference is based on the Potts model [35]. This model provides a mathematical description of interacting spins on a crystalline lattice. Within the model framework, interacting spins are mapped into a graph $G(\eta, \varepsilon)$ [see Figure 1 A)], where each node $i \in \eta = \{1, \ldots, N\}$ has an associated spin a_i,
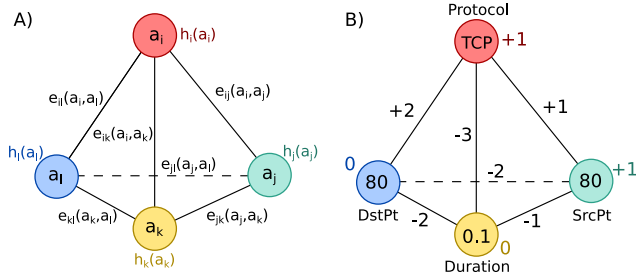
Fig. 1. A) Interacting spins on a crystalline lattice. B) Network flow mapped into a graph structure.

which can assume one value from a set Ω that contains all possible individual quantum states. Each node i has also an associated local field $h_i(a_i)$ that is a function of a_i's state. Meanwhile, each edge $(i,j) \in \varepsilon$, $i,j \in \eta$, has an associated coupling value $e_{ij}(a_i, a_j)$ that is a function of the states of spins a_i and a_j associated to nodes i and j. A specific system configuration has an associated total energy, determined by the Hamiltonian function $\mathscr{H}(a_1 \cdots a_N)$, which depends on all spin states.

In this work, we reuse the intuitions from the Potts model to characterize network flows [see Figure 1 B)]. An individual flow k is represented by a specific graph configuration $G_k(\eta, \varepsilon)$. Instead of spins, each node represents a selected feature $i \in \eta = \{SrcPort, \ldots, Flags\}$. Within a given flow k, each feature i assumes one value a_{ki} from the set Ω_i that contains all possible values for this feature. As in the Potts Model, each feature i has an associated local field $h_i(a_{ki})$. Meanwhile, $\varepsilon = \{(i,j)|i,j \in \eta; i \neq j\}$ is the set of edges determined by all possible pairs of features, creating a fully meshed graph that can represent different flow samples through their common features. Each edge has an associated coupling value determined by the function $e_{ij}(a_{ki}, a_{kj})$.

Since the values of local fields and couplings depend on the values assumed by features within a given flow, each distinct flow will have a different combination of these quantities. As in the Potts Model, the Hamiltonian involving local fields and couplings determines the total "energy" $\mathscr{H}(a_{k1} \cdots a_{kN})$ of each flow. For instance, in Figure 1 B), the total "energy" of the flow is obtained by summing up all values associated with the edges and to the nodes, resulting in a total of -3. Note that what we call energy is analogous to the notion of Hamiltonian in Quantum Mechanics. It is important to note that the model described here is discrete, therefore continuous features must be discretized. The classes for continuous feature discretization are shown in Section V. In the following, we present the framework applied to perform the statistical model inference and subsequent energy-based flow classification.

A. Model Inference

In this section, a statistical model is going to be inferred in terms of couplings and local field values to perform energy-based flow classification. The main idea consists in extracting a statistical model from benign flow samples to infer coupling and local field values that characterize this type of traffic.

When calculating the energies of unlabeled flows using the inferred values, it is expected that benign flows will have lower energies than malicious flows.

Let $(A_1 \cdots A_N)$ be an N-tuple of features, which can be instantiated for flow k as $(a_{k1} \cdots a_{kN})$, with $a_{k1} \in \Omega_1, \ldots, a_{kN} \in \Omega_N$. Each feature value a_{ki} is encoded by an integer from the set $\Omega = \{1, 2, \ldots, Q\}$, i.e., all feature alphabets are the same $\Omega_i = \Omega$ of size Q. If a given feature can only assume M values and $M < Q$, it is considered that values $M+1, \ldots, Q$ are possible, but will never be observed empirically. For instance, if the only possible values for feature *protocol* are {'TCP', 'UDP'}, and given $Q = 4$. In this case, we would have the mapping {'TCP':1, 'UDP':2, ' ':3, ' ':4} and feature values 3 and 4 would never occur.

Now, let $\mathscr{K}$ be the set of all possible flows, i.e., all possible combinations of feature values ($\mathscr{K} = \Omega^N$), and let $\mathscr{S} \subset \mathscr{K}$ be a sample of flows. We can use inverse statistical physics to infer a statistical model associating a probability $P(a_{k1} \cdots a_{kN})$ to each flow $k \in \mathscr{K}$ based on sample $\mathscr{S}$. The global statistical model P is inferred following the Entropy Maximization Principle [36]:

$$\max_P \quad -\sum_{k \in \mathscr{K}} P(a_{k1} \cdots a_{kN}) log(P(a_{k1} \cdots a_{kN})) \quad (1)$$

$$s.t. \sum_{k \in \mathscr{K} | a_{ki} = a_i} P(a_{k1} \cdots a_{kN}) = f_i(a_i) \quad (2)$$

$$\forall i \in \eta; \ \forall a_i \in \Omega;$$

$$\sum_{k \in \mathscr{K} | a_{ki} = a_i, a_{kj} = a_j} P(a_{k1} \cdots a_{kN}) = f_{ij}(a_i, a_j)$$

$$\forall (i,j) \in \eta^2 | i \neq j; \ \forall (a_i, a_j) \in \Omega^2; \quad (3)$$

where $f_i(a_i)$ is the empirical frequency of value a_i on feature i and $f_{ij}(a_i, a_j)$ is the empirical joint frequency of the pair of values (a_i, a_j) of features i and j. Note that constraints (2) and (3) force model P to generate single as well as joint empirical frequency counts as marginals. This way, the model is sure to be coherent with empirical data.

Single and joint empirical frequencies $f_i(a_i)$ and $f_{ij}(a_i, a_j)$ are obtained from set $\mathscr{S}$ by counting occurrences of a given feature value a_i or feature value pair (a_i, a_j), respectively, and dividing by the total number of flows in $\mathscr{S}$. Since the set $\mathscr{S}$ is finite and much smaller than $\mathscr{K}$, inferences based on $\mathscr{S}$ are subjected to undersampling effects. Following the theoretical framework proposed in [34], we add pseudocounts to empirical frequencies to limit undersampling effects by performing the following operations:

$$f_i(a_i) \leftarrow (1-\alpha)f_i(a_i) + \frac{\alpha}{Q} \quad (4)$$

$$f_{ij}(a_i, a_j) \leftarrow (1-\alpha)f_{ij}(a_i, a_j) + \frac{\alpha}{Q^2} \quad (5)$$

where $(a_i, a_j) \in \Omega^2$ and $0 \leq \alpha \leq 1$ is a parameter defining the weight of the pseudocounts. The introduction of pseudocounts is equivalent to assuming that $\mathscr{S}$ is extended with a fraction of flows with uniformly sampled features.

The proposed maximization can be solved using a Lagrangian function such as presented in [36], yielding the

following Boltzmann-like distribution:

$$P^*(a_{k1}\cdots a_{kN}) = \frac{e^{-\mathscr{H}(a_{k1}\cdots a_{kN})}}{Z} \qquad (6)$$

where

$$\mathscr{H}(a_{k1}\cdots a_{kN}) = -\sum_{i,j|i<j} e_{ij}(a_{ki}, a_{kj}) - \sum_{i} h_i(a_{ki}) \qquad (7)$$

is the Hamiltonian of flow k and Z (eq. (6)) is the partition function that normalizes the distribution. Since in this work we are not interested in obtaining individual flow probabilities, Z is not required and, as a consequence, its calculation is omitted. Our objective is to calculate individual flows energies, i.e., individual Hamiltonians as determined in eq. (7).

Note that the Hamiltonian, as presented above, is fully determined regarding the Lagrange multipliers $e_{ij}(\cdot)$ and $h_i(\cdot)$ associated to constraints (2) and (3), respectively. Within the Potts Model framework, the Lagrange multipliers have a special meaning, with the set $\{e_{ij}(a_i, a_j)|(a_i, a_j) \in \Omega^2\}$ being the set of all possible coupling values between features i and j and $\{h_i(a_i)|a_i \in \Omega\}$ the set of possible local fields associated to feature i.

Inferring the local fields and pairwise couplings is difficult since the number of parameters exceeds the number of independent constraints. Due to the physical properties of interacting spins, it is possible to infer pairwise coupling values $e_{ij}(a_i, a_j)$ using a Gaussian approximation. Assuming that the same properties apply for flow features, we infer coupling values as follows:

$$e_{ij}(a_i, a_j) = -(C^{-1})_{ij}(a_i, a_j),$$
$$\forall (i,j) \in \eta^2, \forall (a_i, a_j) \in \Omega^2, a_i, a_j \neq Q \qquad (8)$$

where

$$C_{ij}(a_i, a_j) = f_{ij}(a_i, a_j) - f_i(a_i)f_j(a_j) \qquad (9)$$

is the covariance matrix obtained from single and joint empirical frequencies. Taking the inverse of the covariance matrix is a well known procedure in statistics to remove the effect of indirect correlation in data [37]. Now, it is important to clarify that the number of independent constraints in eq. (2) and eq. (3) is actually $\frac{N(N-1)}{2}(Q-1)^2 + N(Q-1)$, even though the model in eq. (6) has $\frac{N(N-1)}{2}Q^2 + NQ$ parameters. So, without loss of generality, we set:

$$e_{i,j}(a_i, Q) = e_{i,j}(Q, a_j) = h_i(Q) = 0 \qquad (10)$$

Thus, in eq. (8) there is no need to calculate $e_{i,j}(a_i, a_j)$ in case a_i or a_j is equal to Q [34]. Afterwards, local fields $h_i(a_i)$ can be inferred using a mean-field approximation [38]:

$$\frac{f_i(a_i)}{f_i(Q)} = exp\left(h_i(a_i) + \sum_{j,a_j} e_{ij}(a_i, a_j)f_j(a_j)\right),$$
$$\forall i \in \eta, a_i \in \Omega, a_i \neq Q \qquad (11)$$

where $f_i(Q)$ is the frequency of the last element $a_i = Q$ for any feature i used for normalization. It is also worth mentioning that the element Q is arbitrarily selected and could be

replaced by any other value in $\{1\ldots Q\}$ as long as the selected element is kept the same for calculations of the local fields of every feature $i \in \eta$. Note that in eq. (11) the empirical single frequencies $f_i(a_i)$ and the coupling values $e_{ij}(a_i, a_j)$ are known, yielding:

$$h_i(a_i) = ln\left(\frac{f_i(a_i)}{f_i(Q)}\right) - \sum_{j,a_j} e_{ij}(a_i, a_j)f_j(a_j) \qquad (12)$$

In the mean-field approximation presented above, the interaction of a feature with its neighbors is replaced by an approximate interaction with an averaged feature, yielding an approximated value for the local field associated to it.

For further details about these calculations, please refer to [33]. Now that all model parameters are known, it is possible to calculate a given flow energy according to eq. (7). In the following, we are going to present the theoretical framework implementation to perform a two-class, i.e., benign and malicious, flow classification, i.e., Energy-based flow classification (EFC).

B. Energy-Based Flow Classification

The energy of a given flow can be calculated according to eq. (7) based on the values of its features and the parameters from the statistical model inferred in Section IV-A. A given flow energy is the negative sum of couplings and local fields associated with its features, according to a given statistical model. It means that a flow that resembles the ones used to infer the model is likely to be low in energy.

Since EFC is an anomaly-based classifier, the statistical model used for classification is inferred based only on benign flow samples. We would then expect the energies of benign samples to be lower than the energies of malicious samples. In other words, what the energy value of a given flow is capturing is how dissimilar that flow is to a set of known benign flows used to infer the model in the training phase. In terms of frequencies, this means that, if a given flow presents feature values combinations that are very frequent in benign flow samples, its energy will be low. In this sense, it is possible to classify flow samples as benign or malicious based on a chosen energy threshold. The classification is performed by stating that samples with energy smaller than the threshold are benign, and samples with energy greater than or equal to the threshold are malicious. Note that the threshold for classification can be chosen in different ways, and it can be static or dynamic. In this work, we will consider a static threshold.

Algorithm 1 shows the implementation of EFC. In lines 2-5, the statistical model for the sampled flows is inferred, as described by eqs. (4), (5), (8) and (12). Afterward, on lines 6-27, the classifier monitors the network waiting for a captured flow. When a flow is captured, its energy is calculated on lines 9-20, according to the Hamiltonian in eq. (7). The computed flow energy is compared to a known threshold (*cutoff*) value on line 21. In case the energy falls above the threshold, the flow is classified as malicious and should be forwarded to deep packet inspection (line 23) for assessment. Otherwise, the flow is released, and the classifier waits for another flow.

Algorithm 1 Energy-Based Flow Classifier

Input: $benign\_flows_{(K \times N)}$, Q, α, *cutoff*

1: import all model inference functions
2: $f\_i \leftarrow SiteFreq(benign\_flows, Q, \alpha)$
3: $f\_ij \leftarrow PairFreq(benign\_flows, f\_i, Q, \alpha)$
4: $e\_ij \leftarrow Couplings(f\_i, f\_ij, Q)$
5: $h\_i \leftarrow LocalFields(e\_ij, f\_i, Q)$
6: **while** Scanning the Network **do**
7:     $flow \leftarrow$ wait_for_incoming_flow()
8:     $e \leftarrow 0$
9:     **for** $i \leftarrow 1$ to $N - 1$ **do**
10:         $a\_i \leftarrow flow[i]$
11:         **for** $j \leftarrow i + 1$ to N **do**
12:             $a\_j \leftarrow flow[j]$
13:             **if** $a\_i \neq Q$ and $a\_j \neq Q$ **then**
14:                 $e \leftarrow e - e\_ij[i, a\_i, j, a\_j]$
15:             **end if**
16:         **end for**
17:         **if** $a\_i \neq Q$ **then**
18:             $e \leftarrow e - h\_i[i, a\_i]$
19:         **end if**
20:     **end for**
21:     **if** $e \geq cutoff$ **then**
22:         stop_flow()
23:         forward_to_DPI()
24:     **else**
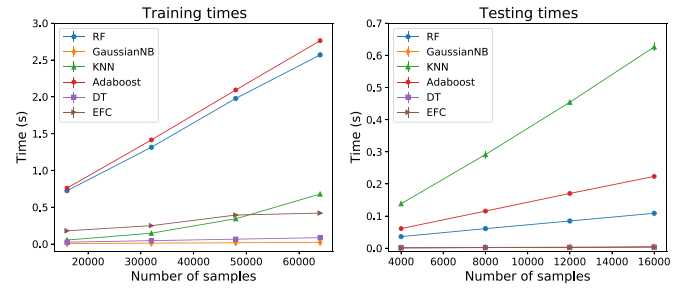25:   release_flow()
26:     **end if**
27: **end while**

Fig. 2. Training and testing runtimes of different ML-based classifiers (scikit-learn implementation) and EFC (Cython implementation) over the CIDDS-001 dataset.

the scikit-learn[2] implementation of the other classifiers in a Ubuntu 20.04 OS using a single thread. The MLP and the SVM classifiers were omitted from the plot for visualization purposes, since their training time complexity is not linear on the number of samples. The results obtained are consistent with the analytical complexity analysis.

Considering the implementation shown in this section, in the following we present the results obtained using EFC and ML-algorithms in classification experiments.

V. RESULTS

In this section, we present the results obtained for EFC and ML-based classifiers in different binary classification experiments considering three different datasets, i.e., CIDDS-001, CICIDS17, and CICDDoS19. First, we show that EFC can separate benign from malicious flows based on their energies, a result that is consistent for all considered datasets. Then, we present EFC's classification performance and compare it to the classification performance of ML-based classifiers in different experiments.

It is important to highlight that the classification experiments we perform in this work were designed not only to assess the performance of different classifiers but also to investigate their capability of adaptation to different domains, i.e., data distributions. Hence, we performed two kinds of experiments: training/testing in the same domain, and training/testing in different domains. For training/testing in the same domain, in each experiment, we assessed the average performance of the classifiers over ten different test sets, containing 10,000 benign and 10,000 malicious samples each, randomly selected from the full dataset. Models were inferred based on 80% the test set and tested on the remaining 20%. The inferred models were used for each experiment to assess the performance of the classifiers over another ten test sets composed of 2,000 benign and 2,000 malicious samples from another domain (data distribution).

A. EFC Characterization

To assess EFC capabilities to separate benign from malicious traffic flow samples correctly, we performed classification experiments considering datasets CIDDS-001, CICIDS17 and CICDDoS19. First, we inferred models based on benign

It is essential to highlight that the time complexity of the training step of EFC is $O((M \times Q)^3 + N \times M^2 \times Q^2)$, where N is the number of samples, M is the number of features, and Q is the size of the alphabet. Meanwhile, the complexity of the classification step for each sample is $O(M^2)$. It means that, in both steps, the complexity is more dependant on the number of features chosen, which can be kept small by using a feature selection mechanism, e.g., Principal Component Analysis (PCA). However, we do not currently explore any feature selection mechanisms because we consider it to be out of scope of this work, in which the main aim is only to present a first version of our newly proposed classifier for NIDS.

EFC has a low training cost, linear in the number of samples (N), when compared to some ML-based classifiers, such as Adaboost (AB), Random Forest (RF), Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP) [39]. EFC's training complexity is considered to be dominated by the term NM^2Q^2 because the number of training samples is expected to be much bigger than both the number of features M and the size of the alphabet Q, which means that the term $(MQ)^3$ is likely not dominant over NM^2Q^2. Figure 2 shows a comparative analysis of training and testing runtimes of different ML-based classifiers and EFC performed on the CIDDS-001 dataset. A Cython implementation of EFC[1] was compared to

---

[1]EFC - https://cyberseclab.gigacandanga.net.br/CyberSecLab/one-class-efc
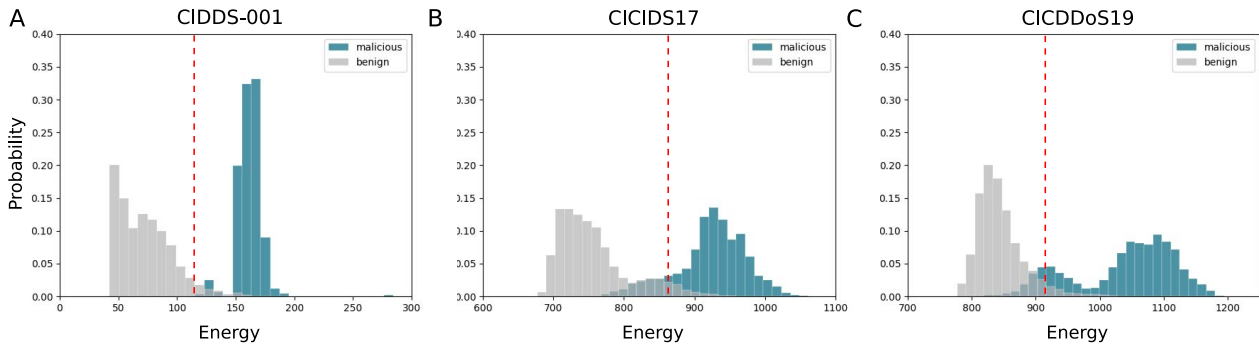
[2]Scikit-learn library - https://scikit-learn.org

Fig. 3. Energy histograms of benign (n = 20,000 in each plot) and malicious (n = 20,000 in each plot) flow samples obtained in the testing phase of a classification experiment performed over the CIDDS-001 dataset (A), CICIDS17 (B) and CICDDoS19 (C) datasets. The energy threshold for classification is shown as a red dashed line and corresponds to the 95th percentile of the energy distribution obtained in the training phase.

samples from the OpenStack (simulated) environment within the CIDDS-001 dataset. This models were used to calculate the energy of different benign and malicious flow samples coming also from the simulated traffic. Figure 3A shows energy values of 40,000 classified flow samples, a merge of the results obtained over ten randomly sampled test sets, as described in the last paragraph of the previous section. The statistical model used to calculate the energies in each test set was inferred based on 8,000 benign flows randomly sampled from the simulated traffic. Flow samples with energy values falling above the energy cutoff, defined as the 95th percentile of the benign traffic training distribution (red dashed line), would be classified as malicious, while the remaining samples would be classified as benign. It is possible to observe that the separation between the two flow classes is clear, i.e., the energy distribution of tested benign flows falls mostly on the left side of the cutoff line, while the energy distribution of tested malicious flows falls mostly on the right side of the cutoff line, as expected.

Figures 3B–C shows the results of analogous experiments performed on the remaining two datasets. Similarly to what is observed for CIDDS-001, when trained on CICIDS17, EFC is also capable of clearly separating the two classes (Figure 3B). This means that the benign energy histogram of tested samples falls mostly on the left side of the cutoff line (95th percentile of the training distribution), while the malicious energy histogram of tested samples falls mainly on the right side of the cutoff line. Again, the same result can be observed for dataset CICDDoS19 (Figure 3C). It is interesting to observe that, although the benign flows energy histograms looks similar regarding variance for the three datasets, the malicious flows energy histogram varies. In CIDDS-001, it has very low variance, reflecting the fact that this dataset contain only four classes of attacks and is highly imbalanced, while in CICIDS17 and CICDDoS19, malicious energy histograms have a broader spread, reflecting the greater variability of malicious flows that exists in those datasets.

The white box nature of the statistical model inferred by EFC is demonstrated in Figure 4, where the energy of different attack classes is broken down to the individual contributions of each pair of features. As shown, for a given attack class,

it is possible to identify which combination of features is contributing the most to its abnormality (red squares) and which pairs of features are similar to normal traffic (blue squares) and might be confounding the model. This analysis was done considering only the couplings and not the local fields. It is interesting to note that different kinds of attacks are characterized by different combinations of abnormal feature pairs, as expected. For instance, the most abnormal thing about DoS attacks is the combination of number of packets and duration, which is consistent with spoofed source address flooding attacks. In turn, Port Scan attacks present an abnormal combination of source and destination ports, in contrast with packets and duration, which are exceedingly normal. Ping Scan attacks are characterized by the protocol ICMP coupled with smaller than usual duration and number of packets, an abnormal combination of source and destination ports, and the absence of flags coupled with a small number of bytes. As for Brute Force, the abnormal coupling between source and destination ports is consistent with unvarying destination and evolving source port, along with abnormally coupled flags and bytes, which reflect short TCP connections.

An analogous breakdown can be done for individual flow samples, allowing for the understanding of which features cause a specific sample to be classified as malicious or benign. This means that the statistical model inferred by EFC allows a direct mapping between specific features of a given flow (model input), e.g., ICMP protocol, and a single packet, and the increase in energy they represent (model output), providing an understanding of *why* a given flow was classified as malicious or benign. Assume, for instance, that EFC classified a flow as an anomaly and the network manager wants to understand the reason for that classification. This is equivalent to understanding *why* that specific flow generated that specific energy value. As the energy of a flow is the sum of the couplings and local fields (functions of the flow features), it is possible to verify which factors of that sum contributed the most to its increase. Consequently, the manager should be able to understand which features contributed the most to classifying the flow as an attack. Now, observing the value assumed by these features in that specific flow, it is possible to clearly understand the classification: some of those features
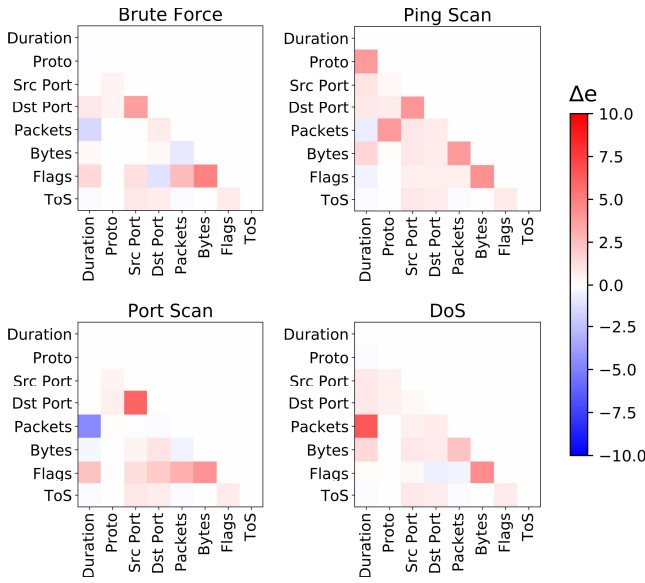
Fig. 4. Individual contributions of each feature pair for the total energy of each attack type within CIDDS-001 dataset: brute force (n = 150), ping scan (n = 100), port scan (n = 800) and DoS (n = 8,950). The heatmap shows the energetic difference Δe that each pair of features has in relation to the average expected energy value of that pair in benign flows (n = 10,000), calculated as $\Delta e = \bar{e}_{attack} - \bar{e}_{benign}$.

are unlikely to appear in benign flows (considering the normal behavior inferred from the training set).

In summary, the results presented in this subsection show that EFC can correctly discriminate between the two flow classes considered, i.e., benign and malicious, consistently across all datasets considered. In addition, it was shown how the total energy of different attack classes can be broken down and analyzed in detail. This is illustrative of the white box nature of the statistical model inferred by EFC. In the following, classification results are shown for different classifiers and compared with the results obtained for EFC.

B. Comparative Analysis of EFC's Performance

We compared EFC to five different ML classifiers: K-Nearest Neighbors (KNN) [40], Decision Tree (DT) [41], [42], Multilayer perceptron (MLP) [43], Naive Bayes (NB) [44], and Support Vector Machine (SVM) [45], all deployed with their default scikit-learn configurations. Additionally, two ensemble methods, namely Adaboost (AB) [46] and Random Forest (RF) [47], were considered, also with default scikit-learn parameters. Flow features were only discretized for EFC (Table V) since discretization would impair the performance of most ML algorithms. The metrics used to compare the results were the F1 score and the area under the ROC curve (AUC). The first metric, F1 score, is the harmonic mean of the Precision and the Recall, i.e.,

$$F1 = \frac{2}{Precision^{-1} + Recall^{-1}} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$
(13)

where $Precision = TP/(TP + FP)$, $Recall = TP/(TP + FN)$, TP are the true positives, i.e., malicious traffic classified as

TABLE V
CLASSES CONSIDERED FOR FEATURE DISCRETIZATION ON CIDDS-001

Feature	List of classes upper limits
Duration	0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.01, 0.04, 1, 10, 100, ∞
Protocol	TCP, UDP, GRE, ICMP, IGMP
Src Port	50, 60, 100, 400, 500, 40000, 60000, ∞
Dst Port	50, 60, 100, 400, 500, 40000, 60000, ∞
Num. of Bytes	50, 60, 70, 90, 100, 110, 200, 300, 400, 500, 700, 1000, 5000, ∞
Num. of Packets	2, 3, 4, 5, 6, 7, 10, 20, ∞
TCP Flags	$\{(f_1, f_2, f_3, f_4, f_5) \mid f_i \in \{0, 1\}\}$

malicious, FP are the false positives, i.e., benign traffic classified as malicious, and FN are the false negatives, i.e., malicious traffic classified as benign. The second metric, the area under the ROC curve (AUC), is one of the most widespread evaluation metrics for binary classifiers [48], [49]. The ROC curve is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. It means that the AUC is the probability that a randomly chosen positive example will receive a higher score than a randomly chosen negative one. One of the main advantages of the AUC is that it is invariant to changes in class distribution, i.e., the ROC curve will not change if the distribution changes in a test set, but the underlying conditional distributions from which the data are drawn stay the same [49], [50]. Since we are interested in evaluating domain adaptation, this metric is particularly interesting to be adopted in this work.

Table V shows the classes considered for feature discretization on CIDDS-001 dataset. Since *TCP Flags* is the discrete feature with more possible values (32 possibilities), the alphabet size Q was set to 32. Each continuous feature values were clustered in a certain number of classes (or bins), up to Q classes. Classes were determined in such a way that the number of values within each class was similar for all classes. Features within datasets CICIDS17 and CICDDoS19 were also discretized in such a way that the number of values within each bin was similar for all bins. These discretizations are not shown here because of the high number of features (80 features) in these datasets.

To evaluate EFC's performance compared to other classifiers, we performed three independent experiments. The first experiment was performed on CIDDS-001. Training was performed on simulated flow samples, while testing was performed on both simulated and real flow samples captured in an external server. The second and the third experiments were cross-dataset experiments performed on CICIDS17 and CICDDoS19. In the former, training was performed on CICIDS17, with testing on both datasets, while in the latter training was performed on CICDDoS19, with testing on both datasets.

Essentially, in each experiment, we measured the performance of the classifiers when trained and tested in the same domain and when trained in one domain and tested in a different one. The performance was measured as the average over ten different test sets, composed of 10,000 benign and 10,000 malicious samples each, randomly selected form the full dataset. With 80% of each test set being used for training and 20% for testing. The test sets containing samples

TABLE VI
AVERAGE COMPOSITION OF EACH OF THE TEST SETS IN EXPERIMENT 1

CIDDS-001 OpenStack		CIDDS-001 real traffic	
Label	Number	Label	Number
normal	10,000	*unknown*	2,000
dos	9,800	*suspicious*	2,000
pingScan	20		
portScan	150		
bruteForce	30		
Total	20,000	**Total**	4,000

TABLE VII
AVERAGE COMPOSITION OF EACH OF THE TEST SETS
IN CROSS-DATASET EXPERIMENTS 2 AND 3

CICIDS17		CICDDoS19	
Label	Number	Label	Number
benign	10,000	*benign*	10,000
FTP Patator	170	*DrDoS DNS*	890
SSH Patator	80	*DrDoS LDAP*	370
DDoS	2,740	*DrDoS MSSQL*	890
PortScan	1,060	*DrDoS NetBIOS*	880
Bot	110	*DrDoS NTP*	890
Infiltration	20	*DrDoS SNMP*	200
Brute force	50	*DrDoS SSDP*	890
SQL injection	10	*DrDoS UDP*	890
XSS	10	*Syn*	890
DoS Hulk	2,730	*TFTP*	890
DoS GoldenEye	2,730	*LDAP*	120
DoS Slowloris	120	*NetBIOS*	140
DoS Slowhttptest	170	*MSSQL*	660
		Portmap	400
		UDP	880
		UDPLag	120
Total	20,000	**Total**	20,000

TABLE VIII
AVERAGE CLASSIFICATION PERFORMANCE AND STANDARD ERROR
(95% CI) - TRAINING PERFORMED ON CIDDS-001 SIMULATED TRAFFIC

Classifier	Train/Test simulated		Train simulated/Test real	
	F1 score	AUC	F1 score	AUC
NB	0.043 ± 0.016	0.502 ± 0.004	0.057 ± 0.024	0.517 ± 0.016
KNN	0.988 ± 0.001	0.994 ± 0.000	0.118 ± 0.014	0.524 ± 0.001
DT	**0.999 ± 0.000**	**0.999 ± 0.000**	0.556 ± 0.007	0.619 ± 0.000
SVM	0.805 ± 0.003	0.951 ± 0.002	0.531 ± 0.005	0.707 ± 0.003
MLP	0.979 ± 0.002	0.993 ± 0.001	0.151 ± 0.016	0.596 ± 0.002
EFC	0.975 ± 0.001	0.997 ± 0.001	**0.675 ± 0.009**	**0.720 ± 0.001**
Ensemble				
AB	0.999 ± 0.000	1.000 ± 0.000	0.594 ± 0.022	0.630 ± 0.000
RF	0.999 ± 0.000	1.000 ± 0.000	0.269 ± 0.018	0.714 ± 0.000

TABLE IX
AVERAGE CLASSIFICATION PERFORMANCE AND STANDARD ERROR
(95% CI) - TRAINING PERFORMED ON CICIDS17

Classifier	Train/Test CICIDS17		Train CICIDS17/Test CICDDoS19	
	F1 score	AUC	F1 score	AUC
NB	0.344 ± 0.049	0.413 ± 0.021	0.344 ± 0.049	0.413 ± 0.049
KNN	0.961 ± 0.001	0.987 ± 0.001	0.457 ± 0.046	0.771 ± 0.001
DT	**0.994 ± 0.001**	**0.994 ± 0.001**	0.168 ± 0.090	0.525 ± 0.001
SVM	0.930 ± 0.003	0.974 ± 0.001	0.264 ± 0.025	0.664 ± 0.003
MLP	0.961 ± 0.003	**0.993 ± 0.001**	0.221 ± 0.033	0.775 ± 0.003
EFC	0.898 ± 0.003	0.975 ± 0.001	**0.787 ± 0.004**	**0.781 ± 0.003**
Ensemble				
AB	0.991 ± 0.002	1.000 ± 0.000	0.228 ± 0.055	0.698 ± 0.002
RF	0.997 ± 0.001	1.000 ± 0.000	0.021 ± 0.003	0.867 ± 0.001

from a different domain were not used for training, hence they were composed of only 2,000 benign and 2,000 malicious samples, randomly selected from the full dataset. EFC's cutoff was defined to be at the 95th percentile of the energy distribution obtained in the training phase based solely in benign samples. It means that we used a completely statistical threshold based only in the training considering benign traffic without adjustments based on malicious samples, such as other ML-Algorithms require. The average composition of the test sets is shown in Tables VI and VII.

Table VIII shows the average performance and standard error (95% confidence interval) of each classifier on the first experiment considering CIDDS-001 dataset. When trained and tested in the same simulated environment, DT is the algorithm presenting the best performance, with an F1-score of 0.999 ± 0.000 and 0.999 ± 0.000 AUC. EFC does also perform well, being the second best in terms of AUC (0.997 ± 0.001). When trained in the simulated environment

and tested in the real environment, EFC outperforms the other classifiers (both simple and ensemble methods) in F1-score (0.675 ± 0.009) and AUC (0.720 ± 0.001). It is noteworthy that all algorithms present a considerable degradation in performance when tested in a different domain, showing how sensitive the inferred models are to changes in data distribution.

Table IX shows the results of experiment two, which was performed on CICIDS17 and CICDDoS19 datasets. When trained and tested on CICIDS17, DT is again the algorithm to present the best performance both in terms of F1-score (0.994 ± 0.001) and AUC (0.994 ± 0.001), though indistinguishable from MLP AUC (0.993 ± 0.001). Notably, when trained on CICIDS17 and tested on CICDDoS19, EFC outperformed the other simple algorithms in both F1-score (0.787 ± 0.004) and AUC (0.781 ± 0.003). Again, it is possible to see that EFC is the best algorithm in adapting to a different data distribution when evaluating both metrics. However, when considering also ensemble methods, RF outperforms EFC, which becomes second best in terms of AUC.

Further, Table X shows the results of experiment three, which was performed on CICIDS17 and CICDDoS19 datasets. Once more, DT overperformed the other classifiers when training and testing on the same dataset, with F1-score of 0.998 ± 0.000 and AUC of 0.998 ± 0.000. When tested on the CICDDoS19 dataset though, EFC achieved the best F1-score (0.641 ± 0.002), while KNN was the best in terms

TABLE X
AVERAGE CLASSIFICATION PERFORMANCE AND STANDARD ERROR
(95% CI) - TRAINING PERFORMED ON CICDDoS19

| | Train/Test CICDDoS19 | | Train CICDDoS19/Test CICIDS17 | |
Classifier	F1 score	AUC	F1 score	AUC
NB	0.590 ± 0.006	0.428 ± 0.007	0.590 ± 0.006	0.428 ± 0.006
KNN	0.960 ± 0.002	0.984 ± 0.001	0.397 ± 0.043	**0.670 ± 0.002**
DT	**0.998 ± 0.000**	**0.998 ± 0.000**	0.259 ± 0.012	0.476 ± 0.000
SVM	0.933 ± 0.002	0.976 ± 0.002	0.239 ± 0.009	0.538 ± 0.002
MLP	0.968 ± 0.002	0.993 ± 0.001	0.227 ± 0.011	0.451 ± 0.002
EFC	0.916 ± 0.002	0.981 ± 0.001	**0.641 ± 0.002**	0.664 ± 0.002
Ensemble				
AB	0.995 ± 0.001	1.000 ± 0.000	0.270 ± 0.013	0.660 ± 0.001
RF	0.997 ± 0.000	1.000 ± 0.000	0.089 ± 0.032	0.623 ± 0.000

of AUC (0.670 ± 0.002). EFC's AUC (0.664 ± 0.002) was the second best, which means that EFC performance was good, taking into consideration both the F1-score and the AUC. Even though this adaptation seems more challenging than the previous ones, EFC's performance was consistent on all the experiments performed.

Taken as a whole, the results presented in this subsection show that EFC is better at adapting to other domains than classical ML-based classifiers on average. In addition to that, it is possible to see that EFC achieves AUC values similar to the best ML algorithms when trained and tested in the same domain, showing that it is capable of performing well even if trained with only half of the information (benign data only) when compared to other classifiers (using malicious and benign data). Not using malicious samples in the training phase is likely to be the reason why EFC is so good at adapting to other domains. EFC's increased capability for domain adaptation when there is a significant difference in data distribution is a highly desirable trait in network flow-based classifiers, since changes in traffic composition are expected to be very frequent, and new kinds of attacks are generated continuously.

Finally, we believe EFC to be an useful tool for network managers, given (i) its more realistic requirements for training (only benign traffic that can be easily captured in the target network), (ii) its adaptability when faced with changes in traffic patterns, and (iii) the possibility to identify which flow features cause a specific network flow to be classified as benign or malicious. However, there is still great room for improvement. One possibility would be to incorporate EFC as a first step of a two-step NIDS, in which the flow samples detected as malicious by EFC would be then sent to deep package inspection. Another possibility is to implement a dynamic threshold that would be adaptable to different network situations, improving classification accuracy. There is also the possibility of performing feature selection previous to model inference, which would greatly reduce the time spent in the model inference phase and possibly also improve classification accuracy. And finally, it would be possible to implement EFC to perform traffic classification at different points in a distributed network. In the following, we present our conclusions and future work directions.

VI. CONCLUSION

In this work, we present a new flow-based classifier for network intrusion detection called Energy-based Flow Classifier (EFC). In the training phase, EFC infers a statistical model based solely on benign traffic samples. Afterward, this statistical model is used to classify network flows in benign or malicious based on "energy" values. Our results show that EFC is capable of correctly performing network flow binary classification considering three different datasets. F1 score (around 97% at best) and AUC (around 99% at best) values obtained using EFC are comparable to the values obtained with other classical ML-based classifiers, such as k-nearest neighbors, decision tree and multilayer perceptron, even though EFC uses only half of the information in the training phase compared to the other algorithms.

In addition to that, we analyzed different classifiers in terms of their capability for domain adaptation and observed that EFC is more suitable to that than classical ML-based algorithms. In all the experiments performed to evaluate that over different datasets, EFC outperformed the other classifiers in F1-score and was among the best ones in AUC. We understand that EFC's capability for domain adaptation is probably linked to the fact that the model inference is based only on benign samples, which helps preventing overfit.

Considering the advantages presented, we believe EFC to be a promising algorithm to perform flow-based traffic classification. Nevertheless, despite the promising results achieved, there is still room for further testing and improvement. In future work, we aim at performing a more comprehensive investigation of EFC's applicability to real-world data and different contexts, such as fraud analysis in bank data. We are already working in a multiclass version of EFC that will be capable of identifying different kinds of known attacks, as well as unknown suspicious flow samples. Finally, we will investigate which improvements can be attained by using a dynamic threshold in EFC and performing feature selection previous to model inference.

REFERENCES

[1] Symantec. (Apr. 2019). *Internet Security Threat Report (ISTR) 2019 Symantec*. [Online]. Available: https://www.symantec.com/security-center/threat-report

[2] M. Pradhan, C. K. Nayak, and S. K. Pradhan, "Intrusion detection system (IDS) and their types," in *Securing the Internet of Things: Concepts, Methodologies, Tools, and Applications*. Hershey, PA, USA: IGI Global, 2020, pp. 481–497.

[3] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Security*, vol. 86, pp. 147–167, Sep. 2019.

[4] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 343–356, 3rd Quart., 2010.

[5] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 159, Jun. 2020, Art. no. 102595.

[6] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Comput. Security*, vol. 70, pp. 238–254, Sep. 2017.

[7] A. Singla, E. Bertino, and D. Verma, "Overcoming the lack of labeled data: Training intrusion detection models using transfer learning," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, 2019, pp. 69–74.

[8] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *Proc. 25th USENIX Security Symp. (USENIX Security)*, 2016, pp. 807–822.

[9] H. Li, Z. Chen, R. Spolaor, Q. Yan, C. Zhao, and B. Yang, "DART: Detecting unseen malware variants using adaptation regularization transfer learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.

[10] M. Zolanvari, M. A. Teixeira, and R. Jain, "Effect of imbalanced datasets on security of industrial IoT using machine learning," in *Proc. IEEE Int. Conf. Intell. Security Informat. (ISI)*, 2018, pp. 112–117.

[11] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.

[12] A. Holzinger, "From machine learning to explainable AI," in *Proc. World Symp. Digit. Intell. Syst. Mach. (DISA)*, 2018, pp. 55–66.

[13] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proc. 16th Eur. Conf. Cyber Warfare Security*, 2017, pp. 361–369.

[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

[15] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Security Technol. (ICCST)*, 2019, pp. 1–8.

[16] R. Vinayakumar, K. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proc. Int. Conf. Adv. Comput. Commun. Informat. (ICACCI)*, 2017, pp. 1282–1289.

[17] A. M. Mahfouz, D. Venugopal, and S. G. Shiva, "Comparative analysis of ML classifiers for network intrusion detection," in *Proc. 4th Int. Congr. Inf. Commun. Technol.*, 2020, pp. 193–207.

[18] S. Khan, E. Sivaraman, and P. B. Honnavalli, "Performance evaluation of advanced machine learning algorithms for network intrusion detection system," in *Proc. Int. Conf. IoT Inclusive Life (ICIIL)*, 2020, pp. 51–59.

[19] X. Tan *et al.*, "Wireless sensor networks intrusion detection based on smote and the random forest algorithm," *Sensors*, vol. 19, no. 1, p. 203, 2019.

[20] J. Kazemitabar, R. Taheri, and G. Kheradmandian, "A novel technique for improvement of intrusion detection via combining random forrest and genetic algorithm," *J. Adv. Defence Sci. Technol.*, vol. 10, no. 3, pp. 287–296, 2019.

[21] T. T. Bhavani, M. K. Rao, and A. M. Reddy, "Network intrusion detection system using random forest and decision tree machine learning techniques," in *Proc. 1st Int. Conf. Sustain. Technol. Comput. Intell.*, 2020, pp. 637–643.

[22] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning," *Procedia Comput. Sci.*, vol. 125, pp. 709–716, Jan. 2018.

[23] M. Ring, D. Landes, and A. Hotho, "Detection of slow port scans in flow-based network traffic," *PloS one*, vol. 13, no. 9, 2018, Art. no. e0204507.

[24] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE Sens. Lett.*, vol. 3, no. 1, pp. 1–4, Jan. 2019.

[25] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving AdaBoost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," in *J. Phys. Conf. Series*, vol. 1192, no. 1, 2019, Art. no. 012018.

[26] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," in *Proc. Int. Symp. Comput. Inf. Sci.*, 2018, pp. 141–149.

[27] J. Li, M. Liu, Z. Xue, X. Fan, and X. He, "RTVD: A real-time volumetric detection scheme for DDoS in the Internet of Things," *IEEE Access*, vol. 8, pp. 36191–36201, 2020.

[28] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9552–9562, Oct. 2020.

[29] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, "Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment," *IEEE Access*, vol. 8, pp. 83765–83781, 2020.

[30] A. AlEroud and G. Karabatis, "A contextual anomaly detection approach to discover zero-day attacks," in *Proc. Int. Conf. Cyber Security*, 2012, pp. 40–45.

[31] D. Plonka, "Flowscan: A network traffic flow reporting and visualization tool," in *Proc. LISA*, 2000, pp. 305–317.

[32] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proc. ICISSP*, 2017, pp. 253–262.

[33] S. Cocco, C. Feinauer, M. Figliuzzi, R. Monasson, and M. Weigt, "Inverse statistical physics of protein sequences: A key issues review," *Rep. Progr. Phys.*, vol. 81, no. 3, Mar. 2018, Art. no. 032601.

[34] F. Morcos *et al.*, "Direct-coupling analysis of residue coevolution captures native contacts across many protein families," *Proc. Nat. Acad. Sci. United States America*, vol. 108, no. 49, pp. E1293–E1301, Dec. 2011.

[35] F. Y. Wu, "The Potts model," *Rev. Modern Phys.*, vol. 54, no. 1, pp. 235–268, Jan. 1982.

[36] E. T. Jaynes, "Information theory and statistical mechanics. II," *Phys. Rev.*, vol. 108, no. 2, pp. 171–190, May 1957.

[37] B. Giraud, J. M. Heumann, and A. S. Lapedes, "Superadditive correlation," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics*, vol. 59, no. 5, pp. 4983–4991, 1999.

[38] A. Georges and J. S. Yedidia, "How to expand around mean-field theory using high-temperature expansions," *J. Phys. A, Math. Gen.*, vol. 24, no. 9, p. 2173, 1991.

[39] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[40] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *J. Royal Stat. Soc. C, Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.

[41] J. R. Quinlan, "Simplifying decision trees," *Int. J. Man Mach. Stud.*, vol. 27, no. 3, pp. 221–234, 1987.

[42] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Trans. Geosci. Electron.*, vol. 15, no. 3, pp. 142–147, Jul. 1977.

[43] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[44] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Proc. Eur. Conf. Mach. Learn.*, 1998, pp. 4–15.

[45] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[46] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. ICML*, vol. 96, 1996, pp. 148–156.

[47] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: A classification and regression tool for compound classification and QSAR modeling," *J. Chem. Inf. Comput. Sci.*, vol. 43, no. 6, pp. 1947–1958, 2003.

[48] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[49] D. Brzezinski and J. Stefanowski, "Prequential AUC: Properties of the area under the ROC curve for data streams with concept drift," *Knowl. Inf. Syst.*, vol. 52, no. 2, pp. 531–562, 2017.

[50] S. Wu, P. Flach, and C. Ferri, "An improved model selection heuristic for AUC," in *Proc. Eur. Conf. Mach. Learn.*, 2007, pp. 478–489.