Spotting Anomalies at the Edge: Outlier Exposure-based Cross-silo Federated Learning for DDoS Detection

Vahid Pourahmadi[1],[2], Hyame Assem Alameddine[3], Mohammad A. Salahuddin[1] and Raouf Boutaba[1]
[1]David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada
[2] Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran
[3]Ericsson Security Research, Montreal, Canada
{v2pourah, mohammad.salahuddin, rboutaba}@uwaterloo.ca
hyame.a.alameddine@ericsson.com

Abstract—Distributed Denial-of-Service (DDoS) attacks are expected to continue plaguing service availability in emerging networks which rely on distributed edge clouds to offer critical, latency-sensitive applications. However, edge servers increase the network attack surface, which is exacerbated with the massive number of connected Internet of Things (IoT) devices that can be weaponized to launch DDoS attacks. Therefore, it is crucial to detect DDoS attacks early, i.e., at the network edge. In this paper, we empower the network edge with intelligent DDoS detection by learning from similarities between different data and DDoS attacks available across the edge servers. To this end, we develop a novel Outlier Exposure (OE)-enabled cross-silo Federated Learning framework, namely FedOE. FedOE enables distributed training of OE-based ML models using a limited number of labeled outliers (i.e., attack flows) experienced at edge servers. We propose a novel OE-based Autoencoder (oAE) that can better discriminate anomalies in comparison to the widely adopted traditional Autoencoder, using a tailored, OE-based loss function. We evaluate oAE in FedOE and demonstrate its ability to generalize to zero-day attacks, with just 50 labeled attack flows per edge server. The results show that oAE achieves a high F1-score for most DDoS attacks, outclassing its non-OE counterpart.

Index Terms—Edge intelligence, federated learning, outlier exposure, anomaly detection, DDoS detection

1 Introduction

Distributed Denial-of-Service (DDoS) attacks continue to plague modern networks with 2.9 million recorded attacks in just the first quarter of 2021 [1], [2], [3]. They disrupt services and impact Quality of Service (QoS) for legitimate users. Emerging networks (e.g., Vehicular Networks, Industrial Internet of Things) rely heavily on distributed edge clouds to offer critical, latency-sensitive services for Internet of Things (IoT) devices [4]. However, the explosion in the number of IoT devices (e.g., 26.4 billion IoT connections are expected by 2026 [5]) significantly increases the attack vector [6]. Adversaries can compromise IoT devices, which often lack sophisticated security mechanisms [7], [8], and harness them as bots to launch DDoS attacks against the geographically distributed edge servers [4], [9], [10]. This has significant implications on the decentralized, and often private network edge data, which plays a quintessential role in the performance of widely adopted Machine Learning (ML)-based anomaly detection [11]. Existing centralized anomaly detection consolidates data from multiple edge servers via the *cloud-centralized* learning paradigm, and falls short in preserving data privacy and entails high communication overhead [11], [12], [13]. In contrast, anomaly detection using local edge server data in the *edge-silo* learning paradigm (though also centralized in its own respect) is less efficient, as it does not harness global knowledge from other edge servers [12].

Cross-silo Federated Learning (FL) is a learning paradigm that leverages distributed training data across the same or different network operators' geo-distributed datacenters. These data cannot be shared due to legal and privacy concerns, nor can be centralized due to their large volume. Therefore, cross-silo FL considers the cooperation of relatively low number of datacenters (i.e., 2-100) to train an ML model, while preserving data privacy and reducing communication overhead [14]. Though FL offers comparable performance to cloud-centralized ML, its performance is highly dependant on whether the model training is supervised or unsupervised [11], [15]. Models trained using supervised ML techniques have been widely adopted in intrusion detection systems [16] due to their high precision in attack detection [17]. However, these models require a large amount of labeled data for training (i.e., benign and anomalous network traffic) and do not generalize to unseen attacks [18], [19]. To overcome the need for labels, unsupervised ML techniques assume that the behavior of benign traffic is starkly different from anomalous traffic. Hence, models trained using unsupervised learning leverage unlabeled data and detect deviations from benign behavior as anomalies, which allows them to generalize to zero-day attacks [18], [19], [20]. Though unsupervised ML alleviates labeling overhead, it leads to a high number of misclassifications, which can severely impact anomaly detection performance.

Misclassifications in anomaly detection is primarily at-

tributed to similarities that may exist between benign and attack traffic, as adversaries often adapt anomalous behavior to be less distinguishable from benign [2]. Indeed, mapping benign data to a region and anomalous data outside that region, as adopted by most unsupervised anomaly detection methods, is challenged by the imprecise boundary between benign and anomalous behavior, which contributes to misclassifications [18]. In our experiments, we also observe a high number of misclassifications using unsupervised anomaly detection methods on a public dataset[1]. This raises a crucial question—*Can we overcome misclassifications in anomaly detection, while ensuring generalization to zero-day attacks?* To answer this question, we resort to semi-supervised learning, which employs a large amount of unlabeled data along with a smaller portion of labeled data to train an ML model [21], thus striking a balance between supervised and unsupervised learning. Most semi-supervised anomaly detection methods use labeled benign training samples, as they are easier to annotate [18], [22]. Other methods leverage labeled attack samples for model training [23], [24]. However, these methods assume the availability of a sizable amount of anomalous samples that cover the majority of the distribution of anomalous behaviors, which is non-trivial to obtain and limits their applicability [18].

In light of the above limitations, we perform analysis on multiple DDoS attacks to better understand the behavior of multiple unsupervised ML models in anomaly detection. In our experiments, we observe that some DDoS attacks (e.g., SYN and UDPLag) exhibit a detection performance that is rather similar across different ML models and learning paradigms (i.e., cross-silo FL and centralized learning). This points to similarities between seemingly very different attacks. For instance, SYN exploits the TCP three-way handshake functionality, whereas UDPLag is carried through a lag switch to monopolize network bandwidth [16]. The Uniform Manifold Approximation and Projection (UMAP)[2] for benign traffic and attacks (i.e., SYN, UDPLag, and UDP) in Figure 1 clearly shows: (i) an overlap between benign traffic and attacks, elucidating the reason behind misclassifications in unsupervised models, (ii) some overlap between SYN and UDPLag attacks, especially over a circular region on the left-side of the benign clusters in Figure 1a and Figure 1b, and (iii) a clear difference between UDPLag and UDP in Figure 1b and Figure 1c, although these attacks exploit the same underlying transport protocol. This raises another important question—*Can similarities in DDoS attacks enhance the performance of anomaly detection?*

To answer this question, we exploit *Outlier Exposure (OE)* [26], which is a semi-supervised technique that leverages limited labeled attack data to better discriminate anomalies. It has proven to be highly efficient for image classification [22], [26], [27], [28]. However, to the best of our knowledge, OE has not yet been used over network traffic for detection of DDoS attacks. The essence of OE lies in determining clear boundaries between benign and attack data by mapping attacks away from the defined benign region [26]. Applying OE for DDoS detection at the network edge is plausible, given that limited labeled attack samples can be made available at each edge server by security experts. However, as edge servers can be subject to attacks that are different from the ones observed during model training, the robustness of an OE-based anomaly detection model against zero-day attacks is a concern that we investigate.

Our main contributions can be summarized as follows:

- We propose a novel OE-based ML model for DDoS attack detection, namely outlier-aware Autoencoder (*oAE*). oAE is a semi-supervised anomaly detection model that is trained to better discriminate benign from anomalous data. It leverages a limited number of outliers and an OE-based loss function, and overcomes high misclassification in unsupervised ML models.
- We develop a novel OE-based cross-silo FL framework for anomaly detection, named *FedOE*. The novelty of FedOE lies within building on similarities between different DDoS attacks experienced across edge servers to enhance the detection performance of oAE. Furthermore, we augment FedOE with a novel federated threshold selection algorithm to maximize the F1-score across the edge servers.
- We evaluate and compare the performance of oAE against its non-OE counterpart (i.e., traditional AE). We show that oAE outperforms AE under different learning paradigms (i.e., cloud-centralized, edge-silo, and cross-silo FL). Further, we show that both models in cross-silo FL, trained using the FedOE, achieve comparable performance to the cloud-centralized learning paradigm, while outperforming the edge-silo learning paradigm.
- We evaluate FedOE and oAE in practice, including the impact of limited number of outlier samples, contaminated training data, and unbalanced data across the edge servers. We show that even a small number of outliers (i.e., only 50 attack flows per edge server) can significantly improve the detection performance of an OE-based model. Furthermore, we show that oAE sustains better against contaminated training data, while being practically unaffected by data imbalance across the edge servers. We also investigate the robustness of oAE to zero-day attacks by exposing it to different subsets of DDoS attacks. The results indicate that training an OE-based model using samples from a DDoS attack (e.g., SYN) can potentially improve the detection of other unseen attacks (e.g., UDPLag).

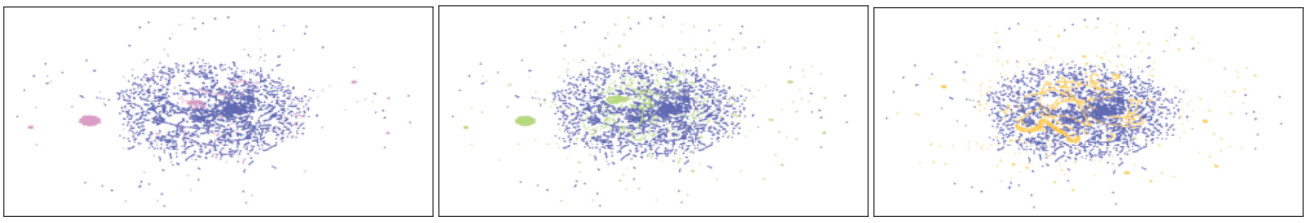
The rest of the paper is organized as follows. In Section 2, we provide a background on centralized and federated ML approaches, and discuss the related works. We present the OE-based anomaly detection models for DDoS detection in Section 3. In Section 4, we delineate the FedOE framework, along with cross-silo FL and the federated threshold selection algorithm. Section 5 exposes the dataset and its preparation in addition to the experimental results and analysis. We conclude in Section 6 and instigate future research directions.

2 Background and Related Works

Attack detection using network traffic has been widely addressed via supervised ML techniques. These techniques

1. We leverage a public dataset [16] in our experiments. The details on the dataset and its preparation for the experiments are available in Section 5.2 and Appendix 7.1.

2. We use UMAP [25] to find and visualize a lower-order (i.e., 2-dimensional) representation of the flow-based features for benign and attack flow samples.



(a) Benign vs. SYN (b) Benign vs. UDPLag (c) Benign vs. UDP

Fig. 1: UMAP's 2-D flow-based features for benign and attacks, i.e., SYN, UDPLag and UDP

build on signatures of the profiled attacks [16], [29], thus limiting detection performance to the learned signatures [18], [19]. To overcome this limitation and generalize to zero-day attacks, unsupervised and semi-supervised ML techniques have been employed for anomaly detection [30]. Nevertheless, the performance of these techniques is highly dependant on data availability and the learning paradigm (i.e., centralized or federated). In this section, we provide a brief overview of these techniques and present the related works.

2.1 Centralized Learning

Centralized learning is the process of training an ML model using data on a training node, collected locally or aggregated from multiple distributed nodes (e.g., edge servers). It has been adopted to train supervised, unsupervised, and semi-supervised ML models including OE-based models. Due to our focus on OE-based ML, we dedicate a specific subsection to it, while contrasting it with other semi-supervised approaches.

2.1.1 Supervised ML

Supervised learning-based attack detection initially employed classical ML techniques. Doshi et al. [31] detected SYN, UDP flood and HTTP GET flood attacks in an IoT environment, using different classifiers, such as k-Nearest Neighbors, Decision Tree (DT), and Random Forest (RF). Similarly, DT and Support Vector Machine (SVM) with linear kernel were employed in [29] for detecting DDoS attacks. Furthermore, Sharafaldin et al. [16] generated the CICDDoS2019 dataset and evaluated the performance of classical ML techniques, including RF, Naïve Bayes, and Logistic Regression, for DDoS detection. Classical ML models achieve reasonable detection performance using a relatively small amount of data. However, their performance relies heavily on feature engineering and they fail to capture complex relationships in input data. Recently, there has been a shift towards Deep Learning (DL) for supervised attack detection. DL is known for its ability to learn complex non-linear relationships in large datasets. For instance, Jia et al. [32] proposed FlowGuard for the detection, identification and mitigation of DDoS attacks in IoT. The authors employed a Long-Short Term Memory (LSTM) model for DDoS attack detection using flow-based features. They also developed a Convolutional Neural Network (CNN) for DDoS attack classification. Although supervised DL has advantages over classical ML techniques, it requires a large amount of labeled data, which is non-trivial to obtain [18].

2.1.2 Unsupervised ML

To overcome the need for labeled data, unsupervised ML has been employed. Choi et al. [33] evaluated different architectures of traditional AE for anomaly detection, and compared it with traditional clustering algorithms. Furthermore, the authors in [19] leveraged AE using sub-flow features (i.e., segment of a flow obtained according to a certain threshold, such as a fixed interval, e.g., 10 ms, or the number of packets) to reduce the response time in detecting DDoS attacks. Their AE performed better in comparison to classical ML algorithms. Time-based features are used in [20], [34] and [35] for anomaly detection using AE and their ensemble, respectively. Evidently, traditional AE seems to be the unsupervised model of choice for numerous works on anomaly detection in network traffic. In fact, AE's ability to profile and reconstruct benign data, while failing to reconstruct unseen anomalous data has proven very efficient for anomaly detection. Therefore, in this paper, we evaluate AE for anomaly detection across numerous, independent DDoS attacks.

2.1.3 Semi-supervised ML

Semi-supervised learning is preferred over supervised learning when the acquisition of a large amount of labeled data is either too expensive or impractical [36]. Semi-supervised learning is a hybrid of supervised and unsupervised learning. For instance, Idhammad et al. [37] developed a sequential semi-supervised ML approach that combines unsupervised learning for anomaly detection using co-clustering algorithm, and supervised ensemble ML classifiers with the Extra-Tree algorithm to classify anomalous traffic. Gao et al. [38] presented a semi-supervised learning approach that combines a fuzziness-based method with an ensemble of the Classification And Regression Tree (CART) to detect network intrusions. However, these works inherit the shortcoming of supervised learning as they require a large amount of attack data for anomaly classification. Semi-supervised learning has also been applied by training an ML model on a large amount of unlabeled data with a small portion of labeled data [36]. The authors in [23] used S4VM, an enhanced version of semi-supervised SVMs, to detect anomalies in network flows. The authors showed comparable performance to supervised methods only when the percentage of labeled training data is increased. Similarly, Xu et al. [24] leveraged One-class SVM for semi-supervised anomaly detection in heterogeneous wireless networks. These works [23], [24] clearly showed that the superior performance of their semi-supervised ML approach for anomaly detection is highly sensitive to the availability

of a sufficiently large amount of labeled data.

2.1.4 Outlier Exposure-based ML

OE was proposed by Hendrycks et al. [26] to improve the performance of semi-supervised anomaly detection, while reducing its sensitivity to the amount of labeled data. The authors argued that leveraging a few Out-Of-Distribution (OOD) (i.e., anomalous) samples during ML model training enables better generalization to unseen anomalies. They showed the benefits of OE in detecting anomalies in natural language processing and image classification. Ruff et al. [22] generalized unsupervised Deep SVDD and proposed Deep SAD for image classification. Deep SAD is a semi-supervised anomaly detection approach, also referred to as unsupervised OE, which utilizes a small pool of labeled anomalous samples during model training. It trains the model to concentrate normal data near a predetermined center, while mapping anomalous samples away from the center. The authors in [27] improved Deep SAD and proposed a HyperSphere Classifier (HSC) as an OE method based on cross-entropy classification. Pang et al. [39] presented an anomaly detection framework that leveraged Deviation Networks and a few labeled anomalies, to enforce a significant deviation in their anomaly scores when compared to normal data. These different OE-based approaches have a similar objective, i.e., ensuring a clear deviation of anomaly scores for anomalous data, to better discriminate between benign and anomalous regions.

Nonetheless, to the best of our knowledge, OE-based ML has not been explored to detect DDoS attacks in network traffic. As AE is one of the most commonly employed neural network (NN) for DDoS detection, in this paper, we propose oAE (i.e., an OE-enabled AE) and evaluate its performance in anomaly detection with respect to various DDoS attacks.

2.2 Federated Learning

FL has been proposed to overcome the limitations of centralized learning, primarily to preserve data privacy and alleviate communication overhead due to data centralization, while maintaining a competitive detection performance [11], [15]. It is also in contrast to distributed learning, which considers centralized data that is partitioned for distributed ML model training across different nodes [40]. In FL, each node (e.g., edge server), also known as FL client, shares the knowledge (i.e., model updates) gained from its local data with a central node, referred to as the FL server, rather than sharing the data itself. The FL server aggregates the model updates from all the FL clients and shares the updated global model parameters with them [14]. Multiple works have adopted FL for anomaly detection in IoT environment. This flavor of FL is known as *cross-device FL*, as it considers a large (i.e., up to 10^{10}) and variable number of IoT devices as FL clients in each training round [14]. For instance, Nguyen et al. [41] leveraged FL to identify compromised IoT devices. In their approach, each IoT gateway trains an IoT device-type specific Gated Recurrent Units (GRU) model that is based on local IoT traffic. The model updates are globally aggregated and pushed to the distributed GRU models. Rahman et al. [12] evaluated cross-device FL for IoT intrusion detection, and compared performance against centralized and self-learning (i.e., learning exclusively on the IoT device) approaches.

In contrast to the widely used cross-device FL, *cross-silo FL* paradigm is adopted when network operators (or organizations) cannot share their data due to legal constraints, or when data belonging to the same operator cannot be centralized between different geographical locations due to communication overhead. Cross-silo FL accounts for a limited number (i.e., 2–100) of stateful FL clients, usually representing different organizations or geo-distributed datacenters that participate in each round of FL training [14]. This makes cross-silo FL suitable for anomaly detection at the network edge, where the limited number of edge servers act as FL clients. Abeshu and Chilamkurti [42] used stacked AE for zero-day attack detection on fog nodes. During each FL training round, fog nodes train the model on a subset of data through parallel training across different threads. The authors showed a higher detection accuracy with a larger number of fog nodes. Kim et al. [43] proposed a FL-based anomaly detection system for IoT. They collaboratively trained a multi-layer perceptron at the edge servers and showed superior performance when compared to models trained in the edge-silo learning paradigm.

Note that none of the aforementioned cross-device FL works accounted for OE-based models. Moreover, works that considered anomaly detection at the network edge, did not discuss the data distribution, nor evaluated its impact on model performance. Unlike these works, we evaluate the impact of different data distributions in cross-silo FL, considering both balanced and unbalanced partitioning of training data across the edge servers, while investigating the advantages of training OE-based models in a federated setting. We highlight that cross-silo FL is rather insensitive to the data distribution in comparison to the highly susceptible edge-silo learning paradigm. More importantly, we depart from an aggregate detection evaluation that conceals performance for individual attack types, and explore the generalization of OE-based anomaly detection models to different unseen DDoS attacks.

3 OE-based Anomaly Detection Model

In this paper, we augment edge servers with intelligent anomaly detection capability. These edge servers leverage the different attacks they experience to facilitate OE-based DDoS detection. In this regard, we leverage NNs, given their ability to learn complex, non-linear relationships in data [35], [44].

3.1 Problem

Given a training dataset $\mathfrak{D} = \{\mathfrak{D}^{\mathrm{u}}, \mathfrak{D}^{\mathrm{a}}\}$ with $\mathfrak{D}^{\mathrm{u}} \cap \mathfrak{D}^{\mathrm{a}} = \emptyset$, where $\mathfrak{D}^{\mathrm{u}}$ is an unlabeled dataset of network traffic (i.e., flows) collected during normal network operation (i.e., $\mathfrak{D}^{\mathrm{u}}$ is benign in majority, but may contain some anomalies), and $\mathfrak{D}^{\mathrm{a}}$ is a labeled dataset encompassing anomalous (i.e., outlier) flows pertaining to DDoS attacks. The objective is to design an OE-based anomaly detection model, the associated loss function and anomaly score function $\zeta(.)$, such that $\zeta(.)$ tries to satisfy $\zeta(u) < \zeta(a) \;\; \forall u \in \mathfrak{D}^{\mathrm{u}}, \forall a \in \mathfrak{D}^{\mathrm{a}}$. $u, a \in \mathbb{R}^d$ are of dimension d (i.e., number of flow-based features) and represent presumably benign and known anomalous flows, respectively.

3.2 oAE: Outlier-aware Autoencoder

A traditional AE is a NN, which has been widely used for anomaly detection. It is composed of an encoder and a decoder. The encoder encodes the input into a latent code z, while the decoder decodes z to reconstruct the input. The difference between the original input and the reconstructed one is quantified as the reconstruction error, e.g., Mean Squared Error (MSE). The reconstruction error corresponds to the *anomaly score* [19]. Once an AE is trained, the reconstruction error is used to classify a new input as benign or anomalous, by comparing it against a predetermined threshold. Typically, a small reconstruction error is associated with benign input, given that an AE, usually trained on benign data, succeeds in reconstructing a similar input. However, anomalous input may exhibit similar behavior to benign input. Therefore, an AE will not always succeed in discriminating benign from anomalous inputs. This leads to a high number of misclassifications.

High misclassifications can be alleviated by forcing the AE to associate starkly higher anomaly score to *known* outliers (i.e., anomalies) during training. This allows to better discriminate benign input from: (i) unknown anomalies, and (ii) stealthy anomalies with inherent behavioral overlap with known outliers. In this vein, we propose a novel OE-based AE, named *oAE*, i.e., a traditional AE augmented with OE capability. oAE is inspired by [22], [27], which proposed OE-based anomaly detection in image classification. oAE has a similar NN structure to traditional AE, along with a modified OE-based loss function. oAE accounts for auxiliary labeled outliers during training, and ensures a high reconstruction error for known outliers and a low reconstruction error for presumably benign inputs, to better discriminate anomalies. Though, in this paper we design and leverage oAE for detecting network traffic anomalies (i.e., DDoS attacks) via network flows, it can be generalized to other inputs.

More formally, we consider a training dataset $\mathfrak{D} = \{\mathfrak{D}^{\mathrm{u}}, \mathfrak{D}^{\mathrm{a}}\}$ as defined in Section 3.1, where $\mathfrak{D}^{\mathrm{a}}$ is a dataset of outliers. Let $\mathfrak{D} = \{(x_1, y_1), ..., (x_q, y_q)\}$, where x_j is a vector of normalized flow-based features for a flow in $\mathfrak{D}$, and $y_j \in \{0, 1\}$ is the label for the flow. Although the flows in $\mathfrak{D}^{\mathrm{u}} \in \mathfrak{D}$ are unlabeled, we assign corresponding $y_j = 0$, as we assume that these flows are benign in majority. In contrast, all flows in $\mathfrak{D}^{\mathrm{a}}$ are anomalous with label $y_j = 1$. Furthermore, the output of the NN is denoted by $\varphi(\mathbf{W}, x_j)$, where $\mathbf{W}$ represents the NN weights that are tuned during the training phase. In this model, the anomaly score assigned to each input x_j is defined as the reconstruction error (1), i.e., :

$$\zeta\left(x_j\right) = \left\| \varphi\left(\mathbf{W}, x_j\right) - x_j \right\|_2^2, \tag{1}$$

where $\|v\|_2$ is the norm two of a vector v. Considering a traditional AE, the loss function (i.e., MSE) is defined as in (2) where $|\mathfrak{D}^{\mathrm{u}}|$ is the number of flows in $\mathfrak{D}^{\mathrm{u}}$.

$$loss(\mathbf{W}, \mathfrak{D}^{\mathrm{u}}) = \frac{1}{|\mathfrak{D}^{\mathrm{u}}|} \sum_{x_j \in \mathfrak{D}^{\mathrm{u}}} \zeta\left(x_j\right) \tag{2}$$

The traditional AE loss function in (2) is geared to minimize the reconstruction error of the benign flows. In contrast, the OE-based loss function of oAE is composed of: (i) an expression that minimizes the reconstruction error of the benign flows (i.e., $\zeta(x_j)\ \forall x_j : y_j = 0$), and (ii) an expression that maximizes the reconstruction error of the outlier flows (i.e., $\zeta(x_j)\ \forall x_j : y_j = 1$). Additionally, inspired by the results in [27], instead of maximization or minimization of $\zeta(x_j)$ directly, we define oAE's OE-based loss function (3) for an input $(x_j, y_j) \in \mathfrak{D}$ as:

$$loss(\mathbf{W}, \mathfrak{D}) = \frac{1}{|\mathfrak{D}|} \sum_{x_j, y_j \in \mathfrak{D}} \Gamma\left(x_j, y_j\right), \tag{3}$$

where

$$\Gamma\left(x_j, y_j\right) = -\Big((1-y_j) \log \rho\left(\zeta(x_j)\right) + y_j \log\left(1 - \rho\left(\zeta(x_j)\right)\right) \Big) \tag{4}$$

and $\rho(x) = \exp\left(-(\sqrt{(x+1)} - 1)\right)$ is a non-increasing function with a value between zero and one. The defined $\rho(x)$ is a Pseudo-Huber function that is also used in robust regression. As discussed in [27], such a selection of $\rho(x)$ yields better results in anomaly detection. oAE is semi-supervised in nature, as it leverages a large number of unlabeled flows along with a limited number of labeled anomalous flows. It benefits from OE, and better generalizes to anomalies that are not seen during training.

4 FedOE: OE-based Cross-silo FL Framework for Anomaly Detection

We combine the benefits of OE in cross-silo FL for DDoS detection at the network edge. To facilitate this, we develop an OE-based cross-silo FL framework for anomaly detection, named *FedOE*. In this section, we briefly present the FedOE framework and cross-silo FL for training an OE-based anomaly detection model $\mathcal{M}$. We also propose a federated threshold selection algorithm, which determines a threshold to discriminate between benign and anomalous network flows.

4.1 FedOE—The Framework

The FedOE framework facilitates the training of an anomaly detection model $\mathcal{M}$ using flow-based features (e.g., flow duration, flag count, sum of packets per flow, etc. (cf., Appendix 7.1.1)) in a federated setting. It is composed of distributed entities, mainly, edge servers belonging to a set C of FL clients, and a FL server $\mathcal{J}$ depicted as the central server in Figure 2. Each edge server $c_i \in C$ monitors and collects corresponding raw traffic using the *network traffic monitoring* module. The latter transfers the collected traffic to the *network traffic preparation* module, which prepares the datasets (cf., Section 5.2) used by $\mathcal{M}$. This module also labels a small number of attack flows, i.e., facilitated by security experts, to enable OE-based anomaly detection. In the absence of security experts or attacks experienced at the edge, this module can be injected with simulated attacks. In addition, as FedOE builds on knowledge sharing between the edge servers, the availability of attack flows at each edge server is not a necessity. The prepared datasets are pre-processed by the *feature engineering* module that performs feature extraction, reduction and normalization. The datasets are then provided to the *anomaly detection* module for OE-based cross-silo FL (cf., Section 4.2).

The central server $\mathcal{J}$ is responsible for cross-silo FL orchestration through the *intelligence orchestration* module.

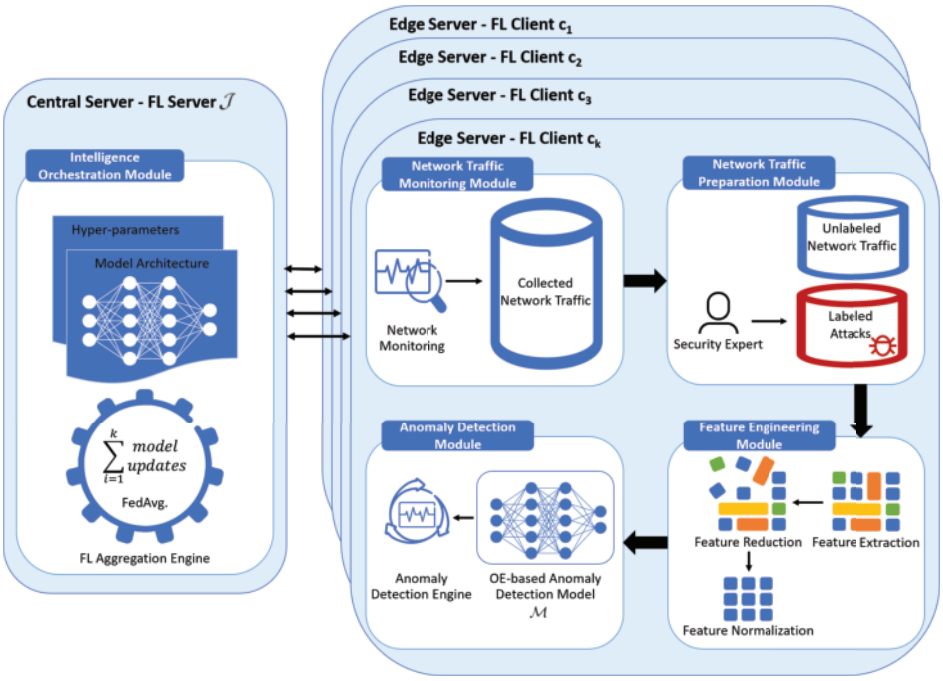


Fig. 2: The FedOE framework

This module shares the structure of $\mathcal{M}$ and model hyper-parameters with each $c_i \in C$. Furthermore, $\mathcal{J}$ includes a *FL aggregation engine* that aggregates the model updates (i.e., NN weights) and selects a threshold (cf., Section 4.3) for anomaly detection. It is worth noting that FedOE is neither bound to a particular anomaly detection model (e.g., oAE that we employ), nor to the flow-based features. FedOE can adapt to other OE-based anomaly detection models and feature sets (e.g., packet or time-based features).

4.2 Cross-silo Federated Learning

The interaction between FedOE entities, i.e., each edge server $c_i \in C$ and central server $\mathcal{J}$, and respective modules is governed by the cross-silo FL process. Cross-silo FL consists of training the OE-based anomaly detection model $\mathcal{M}$ by leveraging data across each edge server $c_i \in C$ (i.e., FL clients), under the orchestration of the FL server $\mathcal{J}$. The training data at each $c_i \in C$ is denoted by $\mathfrak{D}_\mathrm{i} = \{\mathfrak{D}_\mathrm{i}^\mathrm{u}, \mathfrak{D}_\mathrm{i}^\mathrm{a}\}$. $\mathfrak{D}_\mathrm{i}^\mathrm{u}$ and $\mathfrak{D}_\mathrm{i}^\mathrm{a}$ refer to unlabeled (i.e., presumably benign) and labeled anomalous flows, respectively (cf., Section 3.1). The cross-silo FL process includes L training rounds. Typically, in a time-constrained scenario, L is constrained by $\mathcal{M}$'s training time. Otherwise, L should be large enough to ensure that $\mathcal{M}$ adequately captures the network traffic behavior.

The cross-silo FL process begins with $\mathcal{J}$ sending a training request to each $c_i \in C$. The request includes the structure of global $\mathcal{M}$, randomly initialized weights $\mathbf{W}_\ell$ ($\ell \in L;\ \ell = 0$), and hyper-parameters (e.g., batch-size b and number of epochs m per training round). On receipt, each $c_i \in C$ partitions $\mathfrak{D}_\mathrm{i}$ into mini-batches of size b and employs Stochastic Gradient Descent (SGD) to update local models' weights for m epochs, such that the respective loss function (5) is minimized. Note that $loss(\mathbf{W}, \mathfrak{D}_\mathrm{i})$ at each $c_i \in C$ is adapted from (3), where $\mathfrak{D}$ is replaced by the respective c_i's dataset $\mathfrak{D}_\mathrm{i}$. This results in new weights $\mathbf{W}_\ell^i$ at the end of a training round, which are sent to $\mathcal{J}$.

$$\mathbf{W}_\ell^i = \arg\min_{\mathbf{W}} loss(\mathbf{W}, \mathfrak{D}_\mathrm{i}) \tag{5}$$

After receiving the updated weights $\mathbf{W}_\ell^i$ from each $c_i \in C$, $\mathcal{J}$ leverages FedAvg (6) [15] to aggregate the weights and update the global model $\mathcal{M}$:

$$\mathbf{W}_{\ell+1} = \sum_{i=1}^{|C|} \frac{n_i}{n} \mathbf{W}_\ell^i, \tag{6}$$

where n_i and n are the number of flows in $\mathfrak{D}_\mathrm{i}$ and the total flows across all $c_i \in C$ (i.e., $n = \sum_{i=1}^{|C|} n_i$), respectively. $\mathcal{J}$ then shares the new aggregated weights with all $c_i \in C$ at the beginning of a subsequent training round $\ell + 1$. The goal of cross-silo FL is *not to* minimize the loss function for individual $c_i \in C$, but rather to find the global $\mathcal{M}$ weights that minimize the average reconstruction loss across all $c_i \in C$, i.e., $\mathbf{W}$ that minimizes (7), where n_i and n are as previously defined.

$$loss(\mathbf{W}) = \sum_{i=1}^{|C|} \frac{n_i}{n} loss(\mathbf{W}, \mathfrak{D}_\mathrm{i}) \tag{7}$$

4.3 Federated Threshold Selection

Cross-silo FL trains $\mathcal{M}$ for DDoS detection at each edge server $c_i \in C$. As discussed in Section 3, $\mathcal{M}$ assigns an anomaly score to each network flow, where x_j is the flow-based feature vector of the j^{th} flow. To determine if x_j at $c_i \in C$ is anomalous, its anomaly score $\zeta(x_j)$ is compared against a threshold η. If $\zeta(x_j) > \eta$, x_j is classified as anomalous, and benign otherwise. Therefore, the threshold η greatly influences anomaly detection performance.

To determine η, an optimization dataset is often used, which encompasses labeled benign and anomalous flows. With distributed $c_i \in C$, a corresponding optimization

dataset Ω_i *may* be available, which can be leveraged to determine an individual threshold for each $c_i \in C$. However, each individual Ω_i may contain a lesser variety of network flows, which can lead to sub-optimal thresholds for each $c_i \in C$. Though centralizing Ω_i can alleviate this issue, it raises concerns pertaining to privacy and communication overhead.

To improve the performance of $\mathcal{M}$ with the selected threshold, we propose a federated approach for threshold selection that leverages all available Ω_is. Once $\mathcal{M}$ is trained, $\mathcal{J}$ shares the final weights with each $c_i \in C$. Using $\mathcal{M}$, each $c_i \in C$ determines the anomaly score $\zeta(x_j)$ for each x_j in Ω_i. One possible way to deduce the threshold η is to have each c_i share all corresponding $\zeta(x_j)$ with $\mathcal{J}$, which will allow $\mathcal{J}$ to choose the optimal η. However, sharing of $\zeta(x_j)$ is undesirable for privacy concerns.

As an alternate, we allow each $c_i \in C$ to only report limited statistics, mainly, $\max\ \zeta(x_j)$ and $\min\ \zeta(x_j)$ for all anomalous flows in Ω_i. Using these statistics, $\mathcal{J}$ decides on a list of threshold candidates $\eth = \{\eta_1, \eta_2, \cdots, \eta_{\mathcal{Z}}\}$, where η_1 is set to the minimum of the $\min\ \zeta(x_j)$ for anomalous flows from all $c_i \in C$, and $\eta_{\mathcal{Z}}$ is set to the maximum of $\max\ \zeta(x_j)$ from all $c_i \in C$. $\mathcal{J}$ then shares $\eth$ with all $c_i \in C$ and requests the confusion matrices associated with each threshold $\eta_{\mathfrak{z}} \in \eth$. Finally, $\mathcal{J}$ aggregates (i.e., sums) the confusion matrices from all $c_i \in C$ and computes the F1-scores (8) associated with each of them. It then selects $\eta_{\mathfrak{z}}$ that maximizes the F1-score across all $c_i \in C$. The selected $\eta_{\mathfrak{z}}$ is shared with all $c_i \in C$ and used for flagging new anomalous flows.

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

Note that a threshold can be selected with other objectives, such as maximizing the recall, i.e., focusing on not missing any anomalies, or maximizing the precision, i.e., focusing on reducing false alarms. However, similar to [20], we choose to maximize the F1-score as it balances between precision and recall. The federated threshold selection is presented in Algorithm 1.

Algorithm 1 Federated Threshold Selection

Data: Optimization dataset Ω_i (if available) at FL client $c_i\ \forall i = \{0, 1, \cdots, |C| - 1\}$
Results: A threshold η for anomalous flow detection

for $i = 0$ to $|C| - 1$ **do** {FL clients}
  c_i computes $\zeta(x_j)$ for each flow-based feature vector x_j in Ω_i
  c_i sends $\max\ \zeta(x_j)$ and $\min\ \zeta(x_j)$ of all optimization anomalous flows to FL server $\mathcal{J}$
end for
$\mathcal{J}$ sends a list of candidate thresholds $\eth = \{\eta_1, \eta_2, \cdots, \eta_{\mathcal{Z}}\}$ to all $c_i \in C$
for $i = 0$ to $|C| - 1$ **do** {FL clients}
  for $\mathfrak{z} = 1$ to $\mathcal{Z}$ **do** {threshold candidates}
    c_i uses $\eta_{\mathfrak{z}}$ to compute the no. of correct and incorrect decisions for all the flows in Ω_i
  end for
  c_i reports the list of confusion matrices for all threshold candidates in $\eth$ to $\mathcal{J}$
end for
$\mathcal{J}$ sums confusion matrices from all $c_i \in C$ for each $\eta_{\mathfrak{z}}$; computes corresponding F1-scores
$\mathcal{J}$ selects the $\eta_{\mathfrak{z}}$ with the highest F1-score
$\mathcal{J}$ sends the selected $\eta_{\mathfrak{z}}$ to all $c_i \in C$

5 Experiments

5.1 Environment Setup

We consider three FL clients (i.e., edge servers, used interchangeably) and a FL server, to depict a cross-silo FL setting. Each server runs on a separate Virtual Machine (VM) in a cloud environment that is managed via OpenStack[3]. All FL client VMs are powered by 4x Intel Core i7 CPU and 16GB of RAM, while the FL server runs on a VM with 8x Intel Core i7 CPU and 64GB of RAM. As the FL server does not perform computationally intensive tasks, a less powerful VM will also suffice.

We leverage the client/server FL implementation in [45], and modify it for cross-silo FL of non-OE and OE-based anomaly detection models (i.e., $\mathcal{M}$). Initially, the FL server VM awaits the FL client VMs to join. Knowing the IP address/port of the FL server, the FL client VMs send a join message to the FL server. On receipt, the FL server assigns a unique ID to each FL client VM, and shares the structure of $\mathcal{M}$ along with the training hyper-parameters. Other steps of the FL training process follow, such as the update requests and the exchange of updated weights between the FL server and FL clients (cf., Section 4.2). Note that the VMs use common compression techniques on Linux OS to reduce communication overhead of sharing the model weights.

3. https://www.openstack.org/

5.2 Dataset Preparation

A distributed dataset across the different edge servers is important to evaluate FedOE. To the best of our knowledge, datasets that account for benign and anomalous network traffic collected at different edge servers are not publicly available. Therefore, we leverage the CICDDoS2019 dataset [16], which includes various reflection- and exploitation-based DDoS attacks. The wide variety of DDoS attacks (e.g., SYN, UDP, SNMP, SSDP, UDPLag, etc.) available in the dataset, allows us to provide a more holistic evaluation on the efficacy of FedOE across various DDoS attacks. The dataset provides labeled flow-based features (i.e., interchangeably referred to as flows) in CSV format, which are extracted using CICFlowMeter [46].

We distribute the CICDDoS2019 dataset to reflect the decentralized nature of edge servers. We consider the flow destination IP address to distribute the dataset, as flows are typically destined to an edge server that has a group of dedicated IP addresses. However, other distributions based on source IP address, source/destination port, and their combinations, may also be considered. To preserve the validity of analysis, we ensure that all prepared datasets are disjoint. In the following, we briefly describe these datasets. Details of the datasets are available in Appendix 7.1.

- **Training Datasets**: Benign network flows are extracted from the CICDDoS2019 dataset and used to train both non-OE and OE-based ML models. Both balanced distribution (i.e., load balancing is assumed between edge servers) and unbalanced distribution (i.e., edge servers experiencing different loads) of the benign flows across

edge servers are considered for a comprehensive evaluation of FedOE and its non-OE counterparts. Details on the extraction and distribution of the training datasets are available in Appendix 7.1.2.

- **Outlier Datasets**: Outlier datasets complement the training datasets to train the OE-based ML models. As edge servers can be subject to different attacks, we create one outlier dataset per attack type. Each outlier dataset contains network flows from one of the corresponding attack types, i.e., LDAP, MSSQL, SNMP, SSDP, UDPLag, SYN and UDP. The distribution of outlier datasets across edge servers and the number of outliers from each attack type vary based on the experiment (cf., Section 5.5—Section 5.7). Details on the outlier datasets are available in Appendix 7.1.3.
- **Contamination Dataset**: The contamination dataset consists of attack flows, which are used to pollute the purely benign training datasets. It is used to evaluate the robustness of OE-based ML models in the face of contaminated training data, as discussed in Section 5.8.2. Details on the contamination dataset are available in Appendix 7.1.4.
- **Optimization Datasets**: Optimization datasets encompass both benign and attack flows, and facilitate the selection of a threshold to discriminate benign and anomalous flows. Similar to the training datasets, we consider a balanced and an unbalanced distribution of flows across the edge servers. Details on the optimization datasets are available in Appendix 7.1.5.
- **Test Dataset**: The remainder of benign and attack flows in the CICDDoS2019 dataset form the test dataset, which is used to evaluate the anomaly detection performance of non-OE and OE-based ML models. Details on the test dataset are available in Appendix 7.1.6.

The aforementioned datasets facilitate the different experiments performed herein. Their usage is further exposed in the following sections. To ensure transparency and reproducibility of results, we have released the datasets used in this paper to the public.[4]

5.3 Anomaly Detection Model

We evaluate the performance of FedOE using the proposed oAE model with a NN structure of [79, 65, 45, 25, 10, 5, 10, 25, 45, 65, 79]. The first and last numbers in the structure represent the size, i.e., number of neurons in the input and output layers. The remaining numbers correspond to the size of hidden layers. A similar structure is employed for the non-OE counterpart, i.e., traditional AE, which differs in the loss function from oAE. The remaining hyper-parameters and their values for non-OE and OE-based ML models are shown in Table 1. Furthermore, though we evaluated other structures for the NNs as well, we did not notice a significant change in model performance as long as a structure was neither too deep nor too shallow. Therefore, we only report results pertaining to the aforementioned NN structure.

5.4 Evaluation Metrics

To evaluate non-OE and OE-based based ML models independent of a specific threshold, we resort to the Receiver Operating Characteristic (ROC) curves. An ROC curve represents the tuple <True Positive Rate (TPR), False Positive Rate (FPR)> of a ML model for all thresholds. The Area Under the ROC Curve (AUC) is also a common metric used to compare multiple ML models. It provides an aggregated measure of performance across all thresholds. When different test cases are considered to compare multiple ML models, the mean AUC is generally used. An AUC = 1 depicts a perfect model, i.e., with an ideal threshold selection, a ML model can reach a TPR = 1 and a FPR = 0. Furthermore, we also evaluate non-OE and OE-based ML models using a specific threshold. In this regard, we report the F1-scores for various DDoS attacks using the proposed federated threshold selection algorithm (cf., Algorithm 1). Note that the F1-score is highly dependent on the selected threshold. For instance, an ML model with good separative property can result in a low F1-score due to an improper threshold selection.

TABLE 1: Hyper-parameter settings

Hyper-parameter	Value	Hyper-parameter	Value
Training rounds	3000	Weight regularizer	0.1
Learning rate	0.01	Optimizer	Adam
Batch size	2048	Hidden activation	ReLU
Dropout rate	0.1	Output activation	Linear

5.5 FedOE vs. non-OE

5.5.1 Analysis via ROC curves

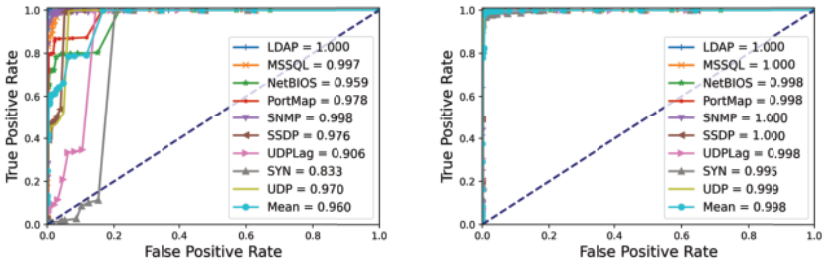
We start by evaluating the performance of non-OE unsupervised anomaly detection model, i.e., traditional AE, denoted FL-AE, which is trained using cross-silo FL with balanced training datasets across the edge servers. Figure 3a highlights the performance of FL-AE via ROC curves for various DDoS attacks. Though respectable with an AUC of over 0.95 for most DDoS attacks and a mean AUC of 0.96, FL-AE under performs for SYN and UDPLag attacks with an AUC of 0.833 and 0.906, respectively.

To alleviate the under performance of unsupervised FL-AE model, we introduce our novel OE-based ML model, oAE (cf., Section 3.2). We evaluate oAE (i.e., in FedOE framework) for DDoS detection at the network edge, and loosely refer it as FedOE. FedOE leverages a small number of outliers (i.e., attack flows) from a few attack types during training, to better discriminate anomalies. It is trained using the balanced training datasets. In this case, the training datasets on the edge servers (i.e., FL clients c_0, c_1 and c_2) are each complemented with 50 flows of LDAP, SSDP, and SYN attacks, respectively, from the outlier datasets. This leads to 150 outliers in total across the three edge servers. As shown in Figure 3b, FedOE achieves a mean AUC of 0.998, which is a significant improvement over its non-OE counterpart (i.e., FL-AE). There is also a considerable improvement in SYN and UDPLag detection with an AUC of 0.995 and 0.998, respectively.

5.5.2 Analysis via F1-scores

Although ROC curves highlight the dominance of an ML model versus another, in practice, the anomaly detection performance is highly dependant on the selected threshold. Therefore, we leverage the federated threshold selection algorithm to select a threshold for flagging anomalous flows across the edge servers. Figure 4 shows the anomaly detection F1-score for various DDoS attacks using FL-AE and

4. The datasets are available for download at: URL will be made available on paper acceptance.



(a) FL-AE (b) FedOE

Fig. 3: ROC curves and AUCs for non-OE and OE-based ML models in cross-silo FL

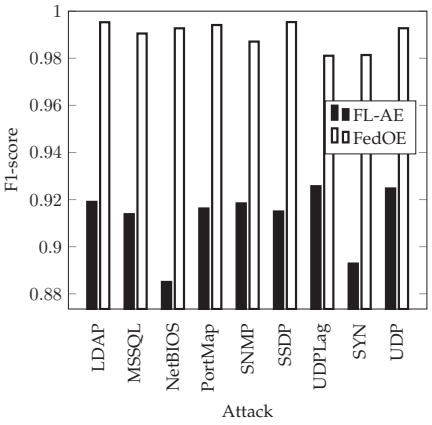


Fig. 4: F1-scores using federated threshold selection (i.e., Algorithm 1) for non-OE and OE-based models

FedOE. For all DDoS attacks, FedOE achieves higher F1-scores in comparison to FL-AE, with an improvement of approximately 6% to 12% across the DDoS attacks. Indeed, an ML model in cross-silo FL is better trained using outliers by learning to enforce a significant deviation of anomaly scores for attack flows. This property, by itself, helps in differentiating between benign and anomalous flows. In addition, it facilitates the selection of a superior threshold, which in turn improves the F1-score for DDoS detection at the network edge.

5.5.3 Discussion

The very high AUCs and F1-scores of FedOE model (i.e., oAE) showcase the advantage of OE-based semi-supervised approach in improving DDoS detection performance. This advantage is partly attributed to cross-silo FL. For instance, a better detection performance is achieved for attacks observed on the edge servers (i.e., used as outliers during model training). Furthermore, as evident in Figure 3 and Figure 4, FedOE generalizes to other attacks (e.g., UDPLag) that are not observed as outliers during model training, i.e., zero-day attacks. This addresses the first question we raised in Section 1, i.e., OE-based ML models overcome misclassifications in unsupervised learning, while generalizing to zero-day attacks. We seek to address the second question in Section 5.6 and Section 5.7.

5.6 FedOE vs. Cloud-centralized vs. Edge-silo

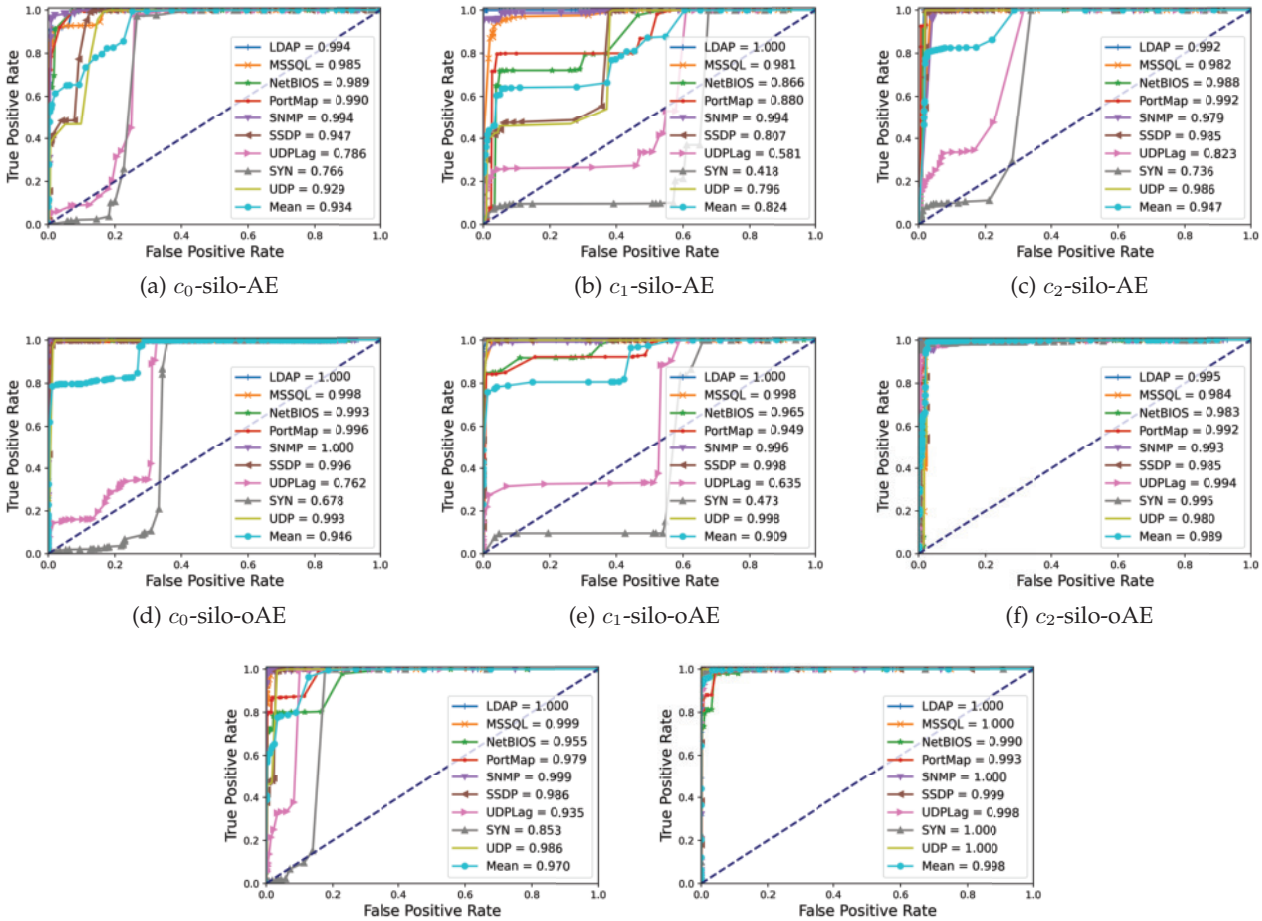
We also compare the performance of traditional AE and oAE under different learning paradigms, as shown in Figure 5. Both ML models leverage the balanced training datasets across the edge servers, while 50 outlier flows from LDAP, SSDP and SYN attacks are introduced on edge servers c_0, c_1 and c_2, respectively. The first and second rows in Figure 5 correspond to traditional AE and oAE, respectively. In each row, we report the AUC for the edge-silo learning paradigm, i.e., each edge server uses local training and outlier (i.e., in the case of oAE) datasets to train its respective ML model. The last row in Figure 5 pertains to the cloud-centralized learning paradigm, where all the training datasets and outliers (i.e., in the case of oAE) are available on a central cloud to train a singleton ML model, which is leveraged by the edge servers for anomaly detection.

In the following, we discuss a few noteworthy observations from Figure 5. Figure 5g and Figure 5h show the performance of AE and oAE in the cloud-centralized paradigm, with a mean AUC of 0.970 and 0.998, respectively. The advantage of OE is attributed to the centralized availability of outliers. Futhermore, comparing Figure 5h to Figure 3b clearly shows that FedOE achieves similar performance to cloud-centralized, while preserving data privacy. Similarly, the benefit of sharing the knowledge of benign flows across edge servers is evident from comparing Figure 3a and Figures 5a to 5c. For instance, Figure 5b shows a very poor performance in detecting SYN attack with an AUC of 0.418, which is significantly improved to 0.833 by using FL in Figure 3a.

The benefit of sharing outliers can be observed by comparing Figure 3b and Figures 5d to 5f. For instance, the c_1-silo model in Figure 5e has an AUC of 0.473 for SYN, which considerably improves to 0.995 in FedOE (cf., Figure 3b). In fact, FedOE uses the knowledge gained from outliers of c_2 to build a more comprehensive model, which is shared between all edge servers. The effect of different outliers is evident by comparing the edge-silo scenarios. For instance, c_0 observes local benign flows and a few outliers from LDAP. Therefore, an improvement in LDAP detection is noticeable from an AUC of 0.994 to 1 in Figure 5a and Figure 5d, respectively. As another example, a significant improvement in AUC of UDPLag can be seen in Figure 5f, where c_2 observes only SYN attacks during oAE training.

Comparing the performance of edge-silo ML models in Figures 5d to 5f with their non-OE counterparts, we see that exposing a few outliers from attack type $\mathcal{A}_i$ has the following effects:

- The model's ability to detect attack type $\mathcal{A}_i$ improves, e.g.,



(a) c_0-silo-AE (b) c_1-silo-AE (c) c_2-silo-AE

(d) c_0-silo-oAE (e) c_1-silo-oAE (f) c_2-silo-oAE

(g) Cloud-centralized-AE (h) Cloud-centralized-oAE

Fig. 5: ROC curves and AUCs for non-OE and OE-based models in edge-silo and cloud-centralized learning

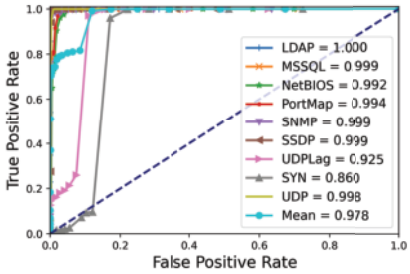
improvement in detecting LDAP, SSDP and SYN attacks in Figures 5d, 5e, and 5f, respectively. This is expected, as the model observes a few outliers (i.e., flows) from these attacks during training. However, the gain from OE is more prominent when the non-OE model struggles in attack detection. For example, the AUC of detecting SYN attack improves from 0.736 (cf., Figure 5c) to 0.995 (cf., Figure 5f) in c_2.

- Exposing outliers from attack type $\mathcal{A}_i$ may improve the detection of another attack $\mathcal{A}_j, j \neq i$. For instance, the AUC of UDPLag improves from 0.823 to 0.994 when c_2 is exposed to SYN attacks (cf., Figure 5c and Figure 5f). This highlights that OE-based models can generalize to attacks that have not been observed during model training. We will reason this further in Section 5.7.
- OE-based edge-silo models may slightly reduce detection performance of some attacks other than $\mathcal{A}_i$. For instance, from Figure 5a and Figure 5d, it can be seen that exposing c_0 to LDAP results in a slight reduction in AUC for SYN and UDPLag attacks. Although OE improves the detection performance on average, we speculate that the model is negatively impacted for attacks that exhibit different statistical behaviour than the observed attack. However, this bias is alleviated with FedOE (cf., Figure 3b), which accumulates the knowledge from the outliers of different edge servers, possibly representing varying attacks.

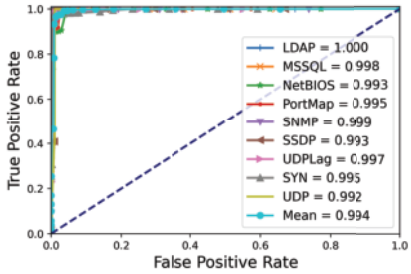
5.7 FedOE with Samples From Different Attacks

Previously, we have shown that edge-silo models have varying performance when they observe different outliers. In this section, we investigate the effect of the outliers available at different edge servers on the detection performance of FedOE. To this end, we still leverage the balanced training datasets across the edge servers, but introduce outliers from LDAP, SSDP, SYN, and UDPLag attacks in varying combinations to c_0, c_1 and c_2. The results of four different combinations are reported in Figure 6. The caption for each subfigure denotes the attacks from which the outliers are introduced. For example, (LDAP, LDAP, SYN) implies that the outliers at c_0 and c_1 are from LDAP, while c_2 is exposed to SYN attack.

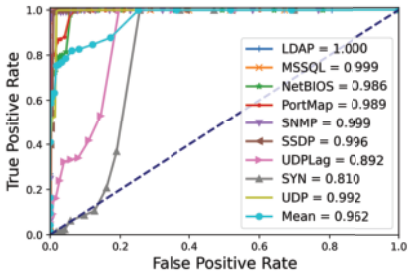
Figure 6a shows the result when SYN flows at c_2 are substituted with LDAP flows, i.e., LDAP, SSDP and LDAP at c_0, c_1 and c_2, respectively. Therefore, Figure 6a is used to highlight the impact of SYN outliers. In comparison to Figure 3b, the AUCs for SYN and UDPLag attacks significantly decrease to 0.86 and 0.925, respectively. Indeed, exposure to SYN outliers (cf., Figure 3b) facilitates FedOE in detecting both SYN and UDPLag attacks with a very high



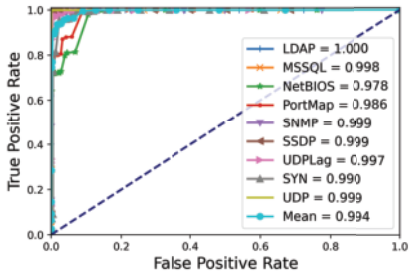
(a) Outliers (LDAP, SSDP, LDAP)



(b) Outliers (LDAP, LDAP, SYN)



(c) Outliers (LDAP, LDAP, LDAP)



(d) Outliers (LDAP, LDAP, UDPLag)

Fig. 6: ROC curves and AUCs with exposure to varying outliers for OE-based model in FedOE

AUC of 0.995 and 0.998, respectively. This is further asserted in Figure 6b with the re-introduction of SYN outliers at c_2, where the AUCs for SYN and UDPLag considerably improve, although SSDP outliers are excluded.

Comparing Figure 6a and Figure 6c highlights the contribution of SSDP outliers on FedOE model performance, where the AUC for detecting SSDP attack slightly decreases to 0.996 in Figure 6c. Interestingly, the SSDP outliers slightly increase the AUC of other attacks, as evident in Figure 6a. Therefore, the DDoS detection performance of a FedOE model improves with exposure to outliers from different attacks, such as LDAP and SSDP. Similarly, we previously observed an improvement in the AUC of UDPLag, when the model is exposed to SYN attack. As shown in Figure 6d, this effect is reciprocal when c_2 observes UDPLag outliers, while the other edge servers observe LDAP. This addresses the second question we raised in Section 1, i.e., OE-based ML models, when exposed to some outliers, capitalize on similarities between different DDoS attacks to improve overall detection performance. These results also show that a network operator can explore similarities (cf., Figure 1) between attacks experienced in the past at different edge servers to boost DDoS detection performance across these servers.

5.8 FedOE Practical Implications

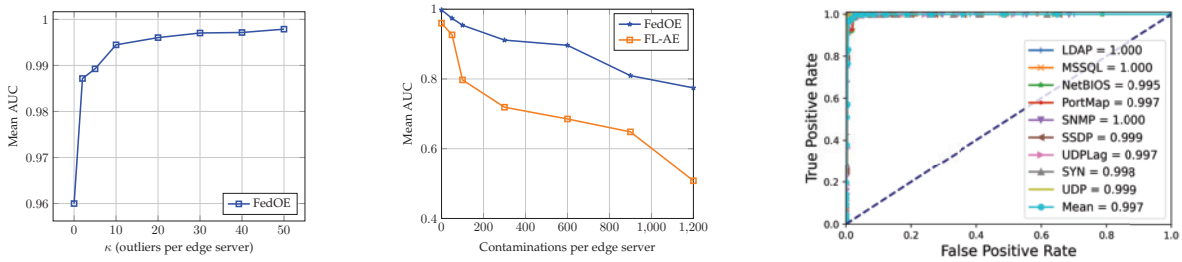
In this section, we evaluate the robustness of FedOE in practical scenarios, when there are limited outliers, the data across the edge servers is contaminated, or unbalanced.

5.8.1 *Impact of the number of outliers:*

In previous experiments, we assumed that each edge server has 50 outliers of a specific attack type to contribute to OE-based ML model training. In this section, we still consider LDAP, SSDP and SYN attacks are observed at c_0, c_1 and c_2, respectively. However, instead of a fixed 50 outliers from each attack, we consider a variable number of outliers across edge servers, denoted κ. Hence, by changing κ, we investigate the impact of the number of observed outliers in FedOE on model performance. The mean AUC across different values of κ is shown in Figure 7. When κ is zero, the mean AUC is 0.960, which corresponds to FL without OE in Figure 3a. However, as the number of outliers increases, the average DDoS detection performance of the model improves, with the highest gain noticeable with the first introduction of outliers.

5.8.2 *Impact of contaminated training datasets*

Ideally, an anomaly detection model should be trained on exclusively benign data collected during normal network operation, i.e., when no attacks are happening, which the ML model leverages to determine the underlying statistics of benign flows. Hence, the purity of collected data is very crucial to model performance, and existence of attacks among benign traffic may mislead the anomaly detection model in estimating the benign flows' distribution. However, in practice, the data collected during network operation may contain attacks. Therefore, in this section, we evaluate the model performance in the face of contaminated data, by introducing the training dataset with a varying number of non-benign (i.e., attack) flows from the contamination dataset. The result is depicted in Figure 7b, where the x-axis corresponds to the number of attack flows contaminating the training dataset at each edge server. As expected, higher contamination reduces the mean AUC of detecting DDoS attacks. However, the performance of FL-AE model deteriorates rapidly with the increase in contamination, which is undesirable in practice. This highlights another benefit of FedOE, where exposure to outliers can better sustain against contamination in the training dataset. For



(a) (b) (c)

Fig. 7: FedOE robustness to: (a) number of outliers (κ), (b) data contamination, and (c) unbalanced datasets

example, contaminating the training data with 600 attacks at each edge server, reduces the mean AUC of FL-AE from 0.96 to 0.69, while the AUC for FedOE drops from 0.99 to 0.89. Although FedOE significantly outperforms FL-AE, it is desirable that FedOE experiences an even lower reduction in mean AUC with an increase in contamination, which is open for future research.

5.8.3 Impact of unbalanced training datasets

The above experiments leverage the balanced training datasets across edge servers. Similar to traditional FL, FedOE is applicable to scenarios where the training data is not equally distributed between the edge servers. Therefore, we also evaluate the performance of FedOE with unbalanced training datasets across edge servers. The result is depicted in Figure 7c, where FedOE effectively shares knowledge from the different edge servers, and achieves a comparable performance (i.e., mean AUC of 0.997) to FedOE with balanced dataset (i.e., mean AUC of 0.998) in Figure 3b. Per attack AUC comparison also asserts similar DDoS detection performance between the two models.

6 Conclusion

DDoS attacks are expected to continue plaguing service availability in emerging networks, which rely on distributed edge servers to provide services with strict QoS guarantees. The edge servers increase the attack surface, making it important to harden the network edge against security threats. In this paper, we empower edge servers with intelligent anomaly detection capability. Through experiments, we unveil the under performance of a popular unsupervised anomaly detection model (i.e., traditional AE) in DDoS detection, leading to a high number of misclassifications. To overcome this shortcoming, we exploit similarities between different types of DDoS attacks and leverage them to enhance the performance of existing anomaly detection. To this end, we adapt an Autoencoder by incorporating OE and enable it to better discriminate benign from attack data in a semi-supervised learning approach.

More precisely, we explore the application of OE for DDoS detection through a novel FedOE framework. FedOE combines cross-silo FL with OE to enhance anomaly detection performance with respect to numerous DDoS attacks. Through extensive experiments, we showcase the superior performance of FedOE with OE-based model, i.e., the proposed oAE, against its non-OE counterpart, i.e., traditional AE. Importantly, the performance gain is prominent with just 50 labeled outlier flows introduced per edge server during OE-based ML model training. In fact, performance improvements are noticeable with merely a few outliers, i.e., 2 labeled attack flows per edge server.

We also explore the impact of the choice of outliers on FedOE performance, and show that, due to similarities between DDoS attacks, observation of outliers from one attack type may help the detection performance of some other attacks, alluding to zero-day attack detection. We also showcase the robustness of FedOE in different scenarios, including contaminated training data and unbalanced data across the edge servers. Although in this paper FedOE was leveraged to exclusively train and evaluate oAE, we would like to emphasize that it can be used to train any OE-based ML model. This paper sheds light on the importance of incorporating knowledge of outliers that might be available at edge servers, while training anomaly detection models. In the future, we will evaluate the performance of OE-based ML models with other features and datasets, and explore ways to further improve detection performance in the face of contaminated training data. One possibility might be to account for unlabeled anomalous flows on top of outliers in the loss function of the OE-based anomaly detection models.

Acknowledgments

The authors would like to thank Dr. Makan Pourzandi, Dr. Stere Preda, Dr. Michael Liljenstam and Dr. Jakob Sternby from Ericsson Research, for their invaluable feedback. This work is supported in part by Ericsson Canada, and in part by the NSERC CRD Grant 536445-18.

References

[1] Carol Hildebrand, "The Beat Goes On," 2021, [Accessed 15-October-2021]. [Online]. Available: https://www.netscout.com/blog/asert/beat-goes

[2] X. Fu and E. Modiano, "Fundamental limits of volume-based network dos attacks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS)*, vol. 3, no. 3, pp. 1–36, 2019.

[3] A. Abhishta, R. van Rijswijk-Deij, and L. J. Nieuwenhuis, "Measuring the impact of a successful ddos attack on the customer behaviour of managed dns service providers," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 5, pp. 70–76, 2019.

[4] T. Madi, H. A. Alameddine, M. Pourzandi, and A. Boukhtouta, "Nfv security survey in 5g networks: A three-dimensional threat taxonomy," *Computer Networks*, vol. 197, p. 108288, 2021.

[5] Ericsson, "Ericsson mobility report," 2021, [Accessed 15-October-2021]. [Online]. Available: https://www.ericsson.com/4a03c2/assets/local/mobility-report/documents/2021/june-2021-ericsson-mobility-report.pdf

[6] K. Doshi, Y. Yilmaz, and S. Uludag, "Timely detection and mitigation of stealthy ddos attacks via iot networks," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[7] I. Cvitić, D. Peraković, M. Periša, and M. Botica, "Novel approach for detection of iot generated ddos traffic," *Wireless Networks*, vol. 27, no. 3, pp. 1573–1586, 2021.

[8] I. Cvitić, D. Perakovic, B. B. Gupta, and K.-K. R. Choo, "Boosting-based ddos detection in internet of things systems," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2109–2123, 2021.

[9] Q. He, C. Wang, G. Cui, B. Li, R. Zhou, Q. Zhou, Y. Xiang, H. Jin, and Y. Yang, "A game-theoretical approach for mitigating edge ddos attack," *IEEE Trans. on Dependable and Secure Computing*, 2021.

[10] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "A data-driven study of ddos attacks and their dynamics," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 648–661, 2018.

[11] W. Y. B. Lim, N. C. Luong, D. T. Hoang *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020.

[12] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Network*, vol. 34, no. 6, pp. 310–317, 2020.

[13] E. Besson, A. Gouget, and H. Sibert, "The gaia sensor: an early ddos detection tool," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 2, pp. 7–8, 2006.

[14] P. Kairouz, H. B. McMahan, B. Avent *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[16] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *IEEE Intl. Carnahan Conf. on Security Technology*, 2019, pp. 1–8.

[17] M. Kravchik and A. Shabtai, "Efficient cyber attacks detection in industrial control systems using lightweight neural networks," *arXiv preprint arXiv:1907.01216*, 2019.

[18] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[19] K. Yang, J. Zhang, Y. Xu, and J. Chao, "Ddos attacks detection with autoencoder," in *IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–9.

[20] M. A. Salahuddin, V. Pourahmadi, H. A. Alameddine, M. F. Bari, and R. Boutaba, "Chronos: Ddos attack detection using time-based autoencoder," *IEEE Trans. on Network and Service Mgmt.*, 2021.

[21] A. Singh, R. Nowak, and J. Zhu, "Unlabeled data: Now it helps, now it doesn't," *Advances in neural information processing systems*, vol. 21, pp. 1513–1520, 2008.

[22] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," *arXiv preprint arXiv:1906.02694*, 2019.

[23] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, and K. Singh, "Flow-based anomaly detection using semisupervised learning," in *Intl. Conf. on Signal Processing and Communication Systems*, 2015, pp. 1–5.

[24] S. Xu, Y. Qian, and R. Q. Hu, "A semi-supervised learning approach for network anomaly detection in fog computing," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[25] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[26] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint arXiv:1812.04606*, 2018.

[27] L. Ruff, R. A. Vandermeulen, B. J. Franks, K.-R. Müller, and M. Kloft, "Rethinking assumptions in deep anomaly detection," *arXiv preprint arXiv:2006.00339*, 2020.

[28] A.-A. Papadopoulos, M. R. Rajati, N. Shaikh, and J. Wang, "Outlier exposure with confidence control for out-of-distribution detection," *Neurocomputing*, vol. 441, pp. 138–150, 2021.

[29] S. Sarraf *et al.*, "Analysis and detection of ddos attacks using machine learning techniques," *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, vol. 66, no. 1, pp. 95–104, 2020.

[30] R. Boutaba, M. A. Salahuddin, N. Limam *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, 2018.

[31] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *IEEE Security and Privacy Workshops*, 2018, pp. 29–35.

[32] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "Flowguard: an intelligent edge defense mechanism against iot ddos attacks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552–9562, 2020.

[33] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *Journal of Supercomputing*, vol. 75, no. 9, pp. 5597–5621, 2019.

[34] M. A. Salahuddin, M. F. Bari, H. A. Alameddine, V. Pourahmadi, and R. Boutaba, "Time-based anomaly detection using autoencoder," in *IEEE International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–9.

[35] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," *Network and Distributed System Security Symposium (NDSS)*, 2018.

[36] B. Hussain, Q. Du, and P. Ren, "Semi-supervised learning based big data-driven anomaly detection in mobile wireless networks," *China Communications*, vol. 15, no. 4, pp. 41–57, 2018.

[37] M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for ddos detection," *Applied Intelligence*, vol. 48, no. 10, pp. 3193–3208, 2018.

[38] Y. Gao, Y. Liu, Y. Jin, J. Chen, and H. Wu, "A novel semi-supervised learning approach for network intrusion detection on cloud-based robotic system," *IEEE Access*, vol. 6, pp. 50 927–50 938, 2018.

[39] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 353–362.

[40] X. Wu, J. Zhang, and F.-Y. Wang, "Stability-based generalization analysis of distributed learning algorithms for big data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 3, pp. 801–812, 2019.

[41] T. D. Nguyen, S. Marchal, M. Miettinen *et al.*, "Dïot: A federated self-learning anomaly detection system for iot," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 756–767.

[42] A. Abeshu and N. Chilamkurti, "Deep learning: the frontier for distributed attack detection in fog-to-things computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[43] S. Kim, H. Cai, C. Hua *et al.*, "Collaborative anomaly detection for internet of things based on federated learning," in *IEEE/CIC International Conference on Communications in China*, 2020, pp. 623–628.

[44] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," *IEEE WOWMOM, CCNCPS Workshop*, 2020.

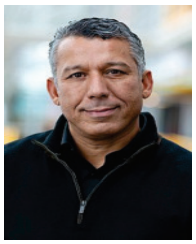
[45] Xuan Luo, "Federated-Learning," 2017, [Accessed 15-October-2021]. [Online]. Available: https://github.com/roxanneluo/Federated-Learning

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

Vahid Pourahmadi received the B.Sc. and M.Sc. degrees from Tehran University, and the Ph.D. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada. He served as a Postdoctoral Fellow at the ECE Department, University of Toronto, Canada, held a Technical Staff position at Blackberry Inc., Ottawa, and was a Wireless Research Engineer at the 5G group of Motorola Mobility, Chicago, IL, USA. His research interests include machine learning, wireless communication, and data analysis.

Hyame Assem Alameddine received her Ph.D. degree in Information and Systems Engineering from Concordia University, Canada in 2019. She holds a Master's degree in Computer Engineering - Information, Systems and Multimedia which she earned in 2015 from Conservatoire National des Arts et des Métiers (CNAM) University, France. She is an experienced researcher at Ericsson, Canada. Before joining Ericsson, she served as a postdoctoral fellow at the University of Waterloo, Canada in 2019 - 2020. She also worked as a programmer and application developer in multiple national and international companies between 2009 and 2014. Her current research interests include Network Function Virtualization, network security, 5G, internet of Things, cloud and edge computing. She serves as a TPC member for international conferences and a reviewer for various journals and magazines.

Mohammad A. Salahuddin received the M.Sc. and Ph.D. degrees in Computer Science from Western Michigan University in 2003 and 2014, respectively. He was a postdoctoral research associate with the Université du Québec à Montréal and University of Waterloo, and a Visiting Scientist with Concordia University. He is currently a Research Assistant Professor of Computer Science at the University of Waterloo. His research interests include the Internet of Things, content delivery networks, network softwarization, network security, and cognitive network management. He serves as a TPC member for international conferences and is a reviewer for various journals and magazines.



Raouf Boutaba received the M.Sc. and Ph.D. degrees in computer science from Sorbonne University in 1990 and 1994, respectively. He is currently a University Chair Professor and the Director of the David R. Cheriton School of Computer science at the University of Waterloo (Canada). He also holds an INRIA International Chair in France. He is the founding Editor-in-Chief of the IEEE Transactions on Network and Service Management (2007-2010) and the current Editor-in-Chief of the IEEE Journal on Selected Areas in Communications. He is a fellow of the IEEE, the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada.