FeCo: Boosting Intrusion Detection Capability in IoT Networks via Contrastive Learning

Ning Wang, Yimin Chen, Yang Hu, Wenjing Lou, Y. Thomas Hou

Virginia Polytechnic Institute and State University, VA, USA

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796926

***Abstract*—Over the last decade, Internet of Things (IoT) has permeated our daily life with a broad range of applications. However, a lack of sufficient security features in IoT devices renders IoT ecosystems vulnerable to various network intrusion attacks, potentially causing severe damage. Previous works have explored using machine learning to build anomaly detection models for defending against such attacks. In this paper, we propose FeCo, a federated-contrastive-learning framework that coordinates in-network IoT devices to jointly learn intrusion detection models. FeCo utilizes federated learning to alleviate users' privacy concerns as participating devices only submit their model parameters rather than local data. Compared to previous works, we develop a novel representation learning method based on contrastive learning that is able to learn a more accurate model for the benign class. FeCo significantly improves the intrusion detection accuracy compared to previous works. Besides, we implement a two-step feature selection scheme to avoid overfitting and reduce computation time. Through extensive experiments on the NSL-KDD dataset, we demonstrate that FeCo achieves as high as 8% accuracy improvement compared to the state-of-the-art and is robust to non-IID data. Evaluations on convergence, computation overhead, and scalability further confirm the suitability of FeCo for IoT intrusion detection.**

I. Introduction

The last decade has seen an exponential growth of the Internet of Things (IoT) devices. Having achieved the milestone of 12 billion connected devices in 2020, it is estimated that by 2025 there will be more than 30.9 billion IoT devices in the market[1]. IoT starts a new paradigm where billions of smart devices with embedded computational capability and Internet connectivity can automatically work with minimal human intervention. Due to low cost and versatility, IoT devices are being used in almost all sectors: healthcare, smart cities, agriculture, and transportation, to name a few [1]–[3].

However, such a pervasiveness also increases the risk of data breaches and cyberattacks. In the past decade, we have seen increased attacks involving IoT devices and IoT systems. Many of the IoT devices have limited on-device resources such as computing power and memory, which limits the amount and types of security mechanisms that can be implemented in them. Many IoT manufactures are not security-savvy. In a rush to roll out new products, very often only minimal security features are included, not to mention providing ongoing support or software security updates. The default configuration of an IoT device usually remains in place if no one makes the effort to change it [4]. One notable attack on IoT is Mirai [5] which overwhelmed several high-profile targets with massive distributed denial-of-service (DDoS) attacks in late 2016. More than half a million devices were infected in a few months. Security patching is one possible remedy to the security issues. However, many devices lack appropriate facilities for automated security updates, or there may be significant delays until device manufacturers provide them. Considering that IoT devices typically connect to the Internet through a local gateway, a more practical and effective idea to secure IoT devices and systems is to implement an IDS in the local gateway. An IDS continuously monitors incoming data streams generated by diverse sources and analyzes them to detect cyber threats.

Ideally, a network IDS device should be placed at a data concentration point in the network for best performance. For instance, most often an IDS device is deployed behind the firewall at the gateway of an edge network. For an IDS in an IoT network, there are two main placement strategies: distributed and centralized. In a distributed placement [6], [7], a local gateway or edge router independently manages its own IDS for the local network. Due to the scarcity of local data, distributed IDS may suffer low accuracy. On the other hand, the rising concern of privacy poses great challenges to a centralized placement [8], [9]. Legal restrictions (e.g., HIPAA[2]) actually prohibit collecting and storing certain types of user data to a central server. In our pursuit to design an efficient, accurate, yet privacy-conscious IDS for IoT network, we resort to the Federated Learning (FL) framework [10]–[12]. FL enables local gateways to cooperatively contribute to the training of a global model by providing their local model parameter to a central server. Through an iterative learning process, the global model achieves good generalization by learning knowledge from a large number of IoT devices. However, due to device heterogeneity (in terms of communication protocols and co-existing technologies), the pattern of normal network traffic varies dramatically among different types of IoT devices. Learning a universal model across all different device types may render the IDS useless. To address this problem, we choose the device-type-specific design [4] that builds an IDS model for each type of IoT devices, which tends to provide more accurate models.

In general, there exist two main approaches for intrusion de-

[1]https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[2]https://www.hhs.gov/hipaa/index.html

tection: anomaly-based Intrusion Detection System (IDS) and signature-based IDS [13]–[16]. An anomaly-based IDS learns a model that represents the normal behavior and generates an alarm when the deviation of an incoming instance from the normal behavior surpasses a security threshold. A signature-based IDS detects intrusions by comparing incoming traffic with the saved signatures in a database of known attacks. Due to the nature of their design, signature-based IDSs are not able to detect zero-day attacks. We focus on anomaly-based IDS in this paper due to its superior capability of detecting novel attacks. With the great success of machine learning in pattern recognition, utilizing machine learning in anomaly-based IDSs is a significant trend in the last two decades. In the machine-learning-based IDSs, it is critical to learn a normal profile. While being able to detect novel attacks, the machine-learning-based IDSs also suffer from the high false-positive rate (FPR) compared to the signature-based IDSs. A false positive occurs when an incoming benign traffic instance deviates from the learned profile and is misclassified as malicious. Due to the large variability of normal network traffic, it has been a real challenge for the machine-learning-based IDSs to find stable notions of normality for such normal traffic.

Extracting the common properties of benign variations precisely and effectively is a key step in learning a stable normal profile. This paper proposes a new way to achieve this— employing contrastive learning [17]–[19] to transform an original traffic instance into a new representation. Specifically, we build a feed-forward artificial neural network (ANN) that takes an original traffic instance as input and outputs a new representation. Our goal is to learn a new feature space where the benign representations lie in a small cluster while attack representations stay far from the benign cluster so that the differentiation of the two can be made easier. By minimizing the volume of a hyper-sphere that encloses the representations of the normal data, we can train an ANN model that is able to extract the common properties of benign variations more precisely, which leads to improved robustness of the normal profile and significantly reduced FPR. Furthermore, in order to reduce the computation overhead, we build a lightweight ANN with only two hidden layers for resource-constrained IoT devices.

Finally, we build **FeCo**, a **Fe**derated-**Co**ntrastive-learning framework, by incorporating FL into the contrastive-learning-based IDS to achieve accurate detection and preserve data privacy simultaneously. In the FeCo design, each local gateway manages a local IDS model and all gateways cooperatively work with a central server to boost local detection performance. To avoid the overfitting of an IDS model to irrelevant features, we propose a two-step feature selection scheme for pre-processing the input data. We remove less significant features and only retain essential information for detection. We extensively evaluate FeCo using a network traffic dataset (i.e., NSL-KDD dataset [20]) to demonstrate the effectiveness of FeCo in detecting intrusions. Our contributions are summarized as follows:

- We propose a novel method for building the "norm" in an anomaly-based IDS by learning new representations for network traffic based on contrastive learning. With the proposed new method, the learned representations of benign inputs lie only in a small cluster, enabling FeCo to extract a stable template for benign inputs. Extensive evaluation results show that representation learning in FeCo significantly boosts its detection accuracy compared to previous works.
- We propose a two-step feature selection scheme to reduce the risk of overfitting. Our feature selection scheme exploits feature correlation and importance and extracts only the essential information for intrusion detection. Such a scheme also helps reduce computation complexity as a result of smaller input dimensionality, making FeCo more suitable for resource-constrained IoT devices.
- We extensively evaluate FeCo using the NSL-KDD dataset and compare it with 11 baselines. The results demonstrate the effectiveness of contrastive-learning-based IDS, showing an 8% accuracy improvement over the state-of-the-art. For zero-day attacks (i.e., attacks unseen by the training dataset), FeCo achieves a recall 8% to 42% higher than other baselines. FeCo is robust to non-IID (Independent and Identically Distributed) data by showing consistent accuracy in different data distributions. We also investigate FeCo on the convergence performance, scalability, and overhead to show that it is suitable for IoT systems.

II. Related Work

We focus on anomaly-based IDSs in this paper. The key component of a general anomaly-based IDS is a model that can represent the legitimate traffic. In what follows, we review the anomaly-based IDSs with a focus on the IDSs in the domain of IoT.

IDS placement is an important design choice in IoT systems compared to computer networks. There exist three IDS placement strategies: distributed IDS placement [6], [7], centralized IDS placement [8], and hybrid IDS placement [9]. Recently, with the advances of Federated Learning (FL) [10], FL-based IDSs [4], [21], [22] are becoming increasingly popular. FL allows a distributed placement to have a better generalization performance as it takes advantage of diverse sets of training data from a large number of IoT devices FL also provides better privacy-preservation in IDSs compared to centralized placement. Nguyen et al. [4] are the first to employ FL in anomaly-based IDSs. In their design, a local gateway uses its local data to train the model and submits the model parameters to the cloud server. The cloud server then aggregates these local models into a global model. Since then, multiple works (e.g., [21] [22]) have explored the use of FL framework to enable decentralized edge devices to learn an anomaly detection model using only on-device data at each edge device.

In the area of anomaly-based IDSs, machine learning mechanisms have been extensively researched in the literature. The most popular strategy for detecting attacks is to monitor a network's activity and report potential abnormal events:

deviations from profiles of normality previously learned from benign traffic [4], [23]–[25]. Du et al. [24] proposed DeepLog that utilizes Long Short-Term Memory to model a system log as a natural language sequence. DeepLog automatically learns log patterns from normal execution and detects anomalies when log patterns deviate from the learned pattern. Mirsky et al. [25] proposed to utilize an ensemble of autoencoders to differentiate between normal and abnormal traffic patterns. The autoencoder reconstructs an input and computes the reconstruction error in terms of root mean squared errors (RMSE). An alarm is generated when RMSE value exceeds a threshold. Nguyen et al. [4] modeled network packets as symbols in a language enabling the use of Gated Recurrent Units (GRU)) for anomaly detection. Specifically, the GRU model estimates a probability of the next symbol, and an alarm is raised if the occurrence probabilities of a sufficient number of packets fall below a detection threshold.

Besides the mechanisms discussed above, some other IDSs directly learn a binary classifier for detecting intrusions. Pajouh et al. [26] proposed a TDTC that utilizes two tires of classification to improve detect rate. TDTC first uses the Naive Bayes classifier to identify anomalous behavior. Then a K-Nearest Neighbor model is used to further detect anomalies from those instances that are classified as normal by the Naive Bayes model. Rathore et al. [27] proposed an ESFCM method that integrates a Fuzzy C-Means with the Extreme Learning Machine (ELM) classifier to achieve efficient attack detection in IoT. Wang et al. [28] exploited the performance of an intrusion detection classifier against evasive intrusions. More machine-learning-based IDSs are discussed in survey papers [29], [30].

III. System Model and Threat Model

We assume an IoT network with multiple types of IoT devices (e.g., IP camera, smart light) connected via a local gateway. As shown in Fig. 1, we focus on the device-type-specific IDS design. For each type of device, a number of gateways cooperatively learn an IDS model through an FL framework. An FL system involves two main components: local gateways (i.e., clients) and a model aggregator, which we describe below.

A **local gateway** $\mathcal{G}$ is a client of the FL system and manages IDSs for detecting compromised IoT devices in the local network. $\mathcal{G}$ may manage multiple IDSs if there are multiple device types present in the local network. $\mathcal{G}$ can choose whether or not to participate in the learning process based on its computation capability, which results in two operation modes: learning mode and consumer mode. $\mathcal{G}$ in learning mode is a client of the FL system, while $\mathcal{G}$ in consumer mode only request an IDS model from a central server. In the rest of the paper, the default mode for $\mathcal{G}$ is learning mode. $\mathcal{G}$ is also responsible for identifying the type of devices when a new IoT device joins the local network. Such device-type-identification techniques were well studied in [31]–[33]. We assume $\mathcal{G}$ to have access to the network traffic of the IoT devices in the local network, which is the same as [4], [25].

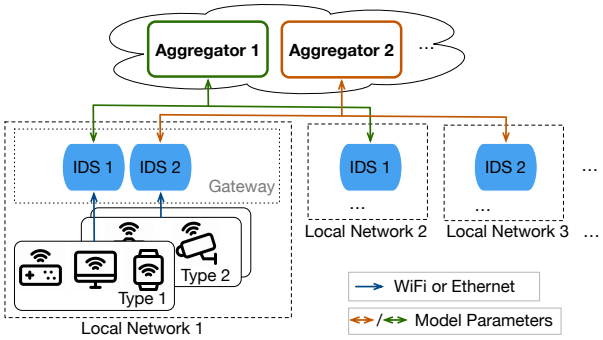


Fig. 1: The system design of our device-type-specific IDS–FeCo. Local gateways cooperatively train an IDS model for each type of IoT devices through FL. The detail of the IDS shown as a blue icon in this figure is depicted in Fig. 3.

A **model aggregator** $\mathcal{A}$ is responsible for aggregating the parameter updates of an IDS model from $\mathcal{G}s$ and sending the up-to-date IDS model to $\mathcal{G}s$. Typically, $\mathcal{A}$ would be run by a service provider such as Amazon, Google, Microsoft, etc.

The FL process between $\mathcal{A}$ and $\mathcal{G}$ can be explained as follows. In the beginning, $\mathcal{A}$ randomly initializes a global model Θ. In each training round, $\mathcal{A}$ selects a subset of clients and distributes the current global model Θ to the selected clients. Each selected client initializes its local IDS model θ by Θ and continues training the model with its local data. In the proposed FeCo system, the local training is based on a contrastive learning algorithm (see Sec. IV). After several local training epochs, $\mathcal{G}$ uploads its model weights θ_i to $\mathcal{A}$. At the end of each round, $\mathcal{A}$ aggregates local models received from $\mathcal{G}s$ following an aggregation rule such as *FedSGD* and *FedAvg* [10]. At the end of a FL task, $\mathcal{A}$ outputs the final global model.

Threat Model: IoT devices are easy targets of many network-based intrusions, such as unauthorized access, address spoofing, false data injection, disruption of network connectivity. Once compromised, they tend to be exploited to launch attacks to other network components or services. In this paper, our goal is to detect malicious network traffic for either purposes.

IV. FeCo Design

A. Workflow of IDS

The workflow of the proposed FeCo is shown in Fig. 2 which provides an overall view of its operations. For model training, (①) a local gateway $\mathcal{G}$ requests initial IDS model parameters according to the types of IoT devices and initializes its IDS model with the received parameters; (②) $\mathcal{G}$ then extracts *significant features* from the raw traffic data and (③) feeds the extracted features into the IDS model; $\mathcal{G}$ further trains the local IDS model using the contrastive learning algorithm and (④) uploads the model parameter update to model aggregator $\mathcal{A}$. Steps ① - ④ repeat until the global model converges. For intrusion detection, (⑤) $\mathcal{G}$ uses the local IDS to detect intrusions while monitoring the network traffic of

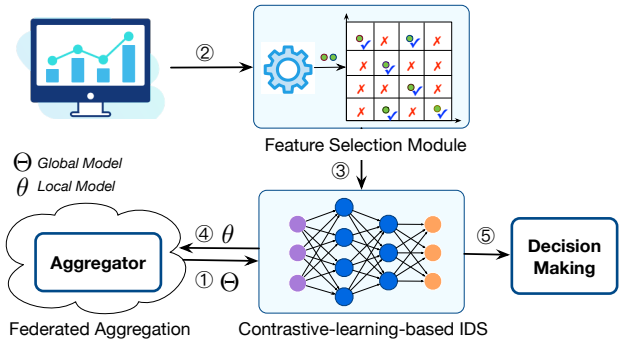


Fig. 2: The workflow of FeCo.

IoT devices. We can see from Fig. 2 that there are three important components in FeCo including *Feature Selection*, *Contrastive-learning-based IDS*, and *Federated Aggregation*. We will introduce the three components in detail respectively in the rest of this section.

B. Feature Selection

In FeCo, We choose to perform feature selection to remove features that are explicitly demonstrated irrelevant to intrusion detection. We propose a two-step feature selection scheme to select essential features for IDS. By removing redundant features, we simplify the model architecture and reduce the training time as well.

1) Redundant Feature Removal: Not all the statistical attributes contain unique information of an individual traffic flow. For example, a feature with zero variance exhibits a constant value in the dataset, and thus it contains no useful information for intrusion detection. Therefore, we first remove the zero-variance features. Furthermore, a feature vector may contain redundant information if there exist multiple highly correlated features. If we use all the highly correlated features for model training, it is likely to cause overfitting because there exists an implicit emphasis on these correlated features. To reduce the risk of overfitting, we use the Spearman rank correlation coefficient [34] to quantify the correlations among the numerical features and then keep only one feature for each set of highly correlated features.

2) Feature Importance Ranking: In FeCo, we propose *feature importance ranking* to rank the importance of features. We employ different methods to measure the importance of categorical features (e.g., the protocol type and service type) and numerical features (e.g., source bytes) since they contribute to the final prediction differently. Specifically, we utilize mutual information (MI) between a feature and the output label to measure the importance of a categorical feature. A zero MI implies that the output label is independent of the target feature. The MI ranges from zero to one, and a higher MI means a higher significance. For numerical features, we employ analysis of variance (ANOVA) to evaluate feature importance. In ANOVA, the observed values of a feature are divided into two groups that are attributable to different values of the label. ANOVA measures whether or not the target feature is statistically different in these two groups. In practice, the ANOVA score is the ratio of the variance between the two groups to the variance within the same group. A larger ANOVA score means higher importance.

The analysis of the proposed feature selection methods and the impact of such data preprocessing on the performance of FeCo are evaluated and reported in Section V-C.

C. Contrastive-learning-based IDS

Contrastive-learning-based IDS is the building block of FeCo. It is deployed for the training process at each $\mathcal{G}$. Contrastive learning is first proposed to improve recognition accuracy in the computer vision field. Our goal of deploying contrastive learning is to trains a model that produces similar representations for all normal traffic instances and make the intrusion representations far from normal representations. We build a binary IDS model and formally introduce Contrastive-learning-based IDS below.

We assume each record in the training dataset consists of two fields: input feature vector $x_i \in \mathbb{R}^d$ and output label $y_i \in \{0, 1\}$ where 0 indicates a normal traffic flow and 1 indicates an intrusion. The goal of the contrastive learning algorithm is to learn a new representation (rather than a label) for each input instance. Specifically, contrastive learning trains an ANN model that takes $x_i \in \mathbb{R}^d$ as input and outputs a new representation $z_i \in \mathbb{R}^o$. The ANN model can be represented by a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^o$ where θ denotes the model parameters. The ANN model of FeCo consists of four layers: the input layer, two hidden layers, and finally the output layer. The corresponding size of each layer is d, 128, 256, and o, respectively. The default value for o is 128.

For convenience, we use $\mathbf{v}_i = f_\theta(x_i)$ to denote the output of a benign input x_i and $\mathbf{u}_i = f_\theta(x_i)$ to denote the output of an intrusion input x_i. We assume the dataset contains N normal traffic flows and M intrusion traffic flows. For each pair of normal inputs, we can obtain representation $\mathbf{v}_i$ and representation $\mathbf{v}_j$. Our goal is to maximize the similarity between $\mathbf{v}_i$ and $\mathbf{v}_j$ and minimize the similarity between $\mathbf{v}_i$ and $\mathbf{u}_m|_{m\in[M]}$ (we define $[M] := \{1, 2, ..., M\}$). To achieve this goal, we define a loss function $\mathcal{L}_{ij}$:

$$\mathcal{L}_{ij} = -log \frac{\exp(\mathbf{v}_i^T \mathbf{v}_j/\tau)}{\exp(\mathbf{v}_i^T \mathbf{v}_j/\tau) + \sum_{m=1}^{M} \exp(\mathbf{v}_i^T \mathbf{u}_m/\tau)}, \quad (1)$$

where $\tau \in [0, 1]$ denotes a temperature parameter, and $\frac{\exp(\mathbf{v}_i^T \mathbf{v}_j/\tau)}{\exp(\mathbf{v}_i^T \mathbf{v}_j/\tau)+\sum_{m=1}^{M} \exp(\mathbf{v}_i^T \mathbf{u}_m/\tau)}$ represents the likelihood that $\mathbf{v}_i$ is close to $\mathbf{v}_j$. $\mathcal{L}_{ij}$, the loss function, is the negative logarithm of the likelihood function. Similarly, we can obtain $\mathcal{L}_{ji}$. The overall loss function of the pair $\mathbf{v}_i$ and $\mathbf{v}_j$ is the summation of $\mathcal{L}_{ij}$ and $\mathcal{L}_{ji}$. We then sum up the loss functions for all pairs of normal vectors and obtain a loss function $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathcal{L}_{ij} + \mathcal{L}_{ji}. \quad (2)$$

We use a stochastic gradient descent (SGD) optimizer to minimize the loss function $\mathcal{L}$. By minimizing the loss function,

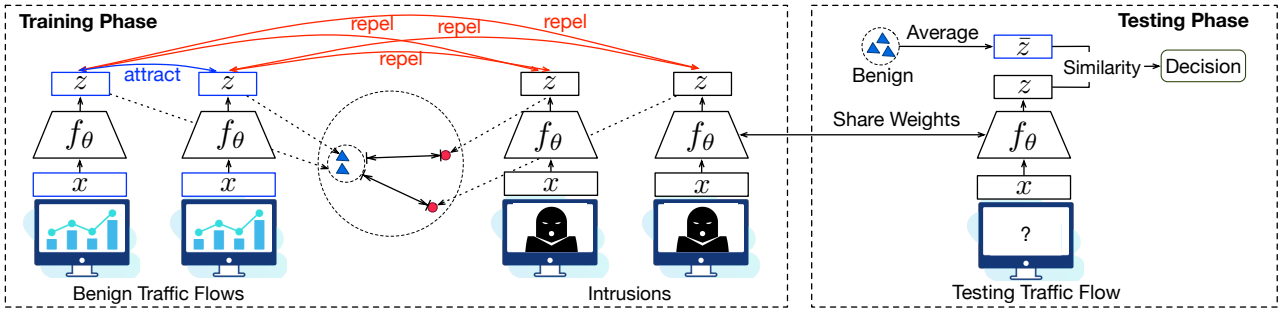


Fig. 3: The training process and testing process of the contrastive-learning-based IDS.

we can achieve the goal of maximizing the similarity among normal representations and minimizing the similarity between normal representations and intrusion representations.

We intuitively interpret the learning process in Fig. 3 as well. We can see that the normal zs (i.e., $\mathbf{v_i}$) attracts each other while they repel intrusion zs (i.e., $\mathbf{u_j}$) in the training process. After several iterations, we can learn a model f_θ that outputs a new representation for each input instance. In the new feature space, the representations of benign traffic flows are expected to fall into a compact cluster, while those of intrusion traffic flows are far from such a cluster. In the testing phase, we collect the representations of benign traffic flows and use the average value of the normalized representations as the normal template $\bar{z}$:

$$\bar{z} = \frac{1}{\sum_i \mathbb{1}(y_i = 0)} \sum_i (\mathbb{1}(y_i = 0) \frac{f_\theta(x_i)}{\|f_\theta(x_i)\|_2}), \tag{3}$$

where $\mathbb{1}(.)$ is the indicator function, and $\mathbb{1}(y_i = 0)$ equals 1 if $y_i = 0$, $\|.\|_2$ denotes the L^2-norm function and $\frac{f_\theta(x_i)}{\|f_\theta(x_i)\|_2}$ represents a normalized representation. After obtaining the normal template $\bar{z}$, we utilize the cosine similarity estimator to measure the similarity $S(x_j^{test})$ between an upcoming traffic flow x_j^{test} and the normal template:

$$S(x_j^{test}) = \frac{\bar{z}^T f_\theta(x_j^{test})}{\|\bar{z}\| \times \|f_\theta(x_j^{test})\|}. \tag{4}$$

The similarity score $S(x_j^{test})$ ranges from 0 to 1. We need a threshold score $0 \leq \rho \leq 1$ for determining whether x_j^{test} is an anomaly or not. In our paper, we obtain the threshold ρ by calculating the statistics of the scores of benign training data. Particularly, we first sort the scores of benign data in ascending order and obtain the sorted sequence $\mathbf{S} = [S_1, S_2, ..., S_N]$. Then we select the p-th percentile of $\mathbf{S}$ as the threshold ρ. In FeCo, we first calculated the index r of the score to select:

$$r = \left\lfloor \frac{p}{100} * N \right\rfloor, \tag{5}$$

where $\lfloor . \rfloor$ denotes the round down function, and $0 \leq p \leq 100$ represents a percentage number. We obtain $\rho = S_r$ which is the r-th entry of $\mathbf{S}$. We can manually select a value for p. We should select a small value for p (e.g., $p = 5$) as a larger p leads to a higher FPR. The final decision $\hat{y}_j$ is made by

$$\hat{y}_j = \mathbb{1}(S(x_j^{test}) < \rho). \tag{6}$$

An input instance is predicted as an intrusion if its similarity score is smaller than threshold ρ.

D. Federated Aggregation

We build FeCo by incorporating the contrastive-learning-based IDS into the federated learning framework. In that case, each client participates in the FL process by providing its model parameter update. We utilize the FedAVG [10] algorithm to aggregate the updates from multiple clients. In time step t, the model aggregator $\mathcal{A}$ computes the global model parameter Θ_t by:

$$\Theta_t = \Theta_{t-1} + \sum_i c_i * (\theta_i - \Theta_{t-1}), \tag{7}$$

where θ_i is the local model parameters at client i and c_i is a weight coefficient. In our paper, c_i based on the size of local training dataset at client i. Particularly, we define c_i as the ratio of the size of the local training dataset at client i to the number of total training samples at all selected clients.

V. Experimental Results

A. Datasets and Experiment Settings

We implement FeCo in the PyTorch platform [35]. We ran all the experiments on a server equipped with an Intel Core i7-8700K CPU 3.70GHz×12, a GeForce RTX 2080 Ti GPU, and Ubuntu 18.04.3 LTS. We experiment with FeCo using the network traffic dataset, NSL-KDD. The NSL-KDD dataset is widely used for IoT scenarios [27], [29], [36] due to lack of dedicated datasets in IoT.

The NSL-KDD dataset [20] includes benign traffic and four categories of intrusions, i.e., DoS, Probing, Remote-to-Local (R2L), and User-to-Root (U2R). Each category of intrusion contains several sub-classes as shown in TABLE I. The whole dataset includes a training set and a testing set, and the training set contains 125,973 records while the test set 22,544 records. Note that some intrusion sub-classes exist only in the testing set, i.e., they are unseen in the training set (e.g., mscan, sqlattack), which makes it possible to evaluate FeCo against zero-day attacks. Each record consists of 41 attributes extracted from a traffic flow and a label indicating its category (i.e., Normal, DoS, Probing, R2L, or U2R). In practice, it is easy to extract attributes from network packets by using existing packet analyzers (e.g., WireShark[3]).

[3]https://www.wireshark.org/

TABLE I: Classes of Intrusions and sub-classes in the NSL-KDD Dataset.

	DoS (10)	Probe (6)	R2L (16)	U2R (7)
Attacks in both Training & Testing Set	back, land, Neptune, pod, smurf, teardrop	ipsweep, nmap, portsweep, satan	ftp_write, guesspasswd, imap, multihop, phf, warezmaster	bufferoverflow, loadmodule, perl, rootkit
Attacks only in Testing Set	apache2, mailbomb, processtable, udpstorm	mscan, saint	httptunnel, named, sendmail, snmpgetattack, snmpguess, xlock, xsnoop, worm	ps, sqlattack, xterm
Attacks only in Training Set			spy, warezclient	

Some papers [37], [38] reported detection accuracy as high as 99% on the NSL-KDD dataset. However, these works introduce a new splitting on the dataset. They either combine the training set and testing set into one then randomly split it into two sets for training and testing, or directly split the training set into two sets. Such arrangements can only demonstrate the effectiveness of IDS models in detecting known intrusions but not unseen intrusions. In this paper, we train our IDS model with the training set and evaluate it using the testing set to explicitly show its performance in detecting unseen intrusions.

Other default settings of FeCo are shown as follows. In order to better simulate the distributed characteristics of a real FL system, we choose 50 clients which is relatively larger than 10 and 15 used in [4], [21]. We generate data for clients by splitting the whole dataset. Therefore, the number of clients and the size of local data would be inversely proportional to each other. By using different splitting strategies, we obtain both IID data and non-IID data (See Sec. V-E for detail). For data preprocessing, we first use our feature selection scheme to remove 10 attributes from the 41 attributes. Then we use the one-hot encoding method to map the remaining 31 attributes into 112 input features. The size of the input layer and the output layer of the ANN model in FeCo are $d = 112$ and $o = 128$, respectively. The machine learning baselines used for comparison are imported from scikit-learn [39]. In FeCo, each client uses an SGD optimizer with a learning rate of 0.001 for local training. The clients perform four epochs of local training before sending its model parameters to $\mathcal{A}$ in each FL round. We set the number of total FL rounds as 15.

B. Evaluation Metrics

For a binary detection problem, there are four important terms: True Positive (TP) means correctly detected as an intrusion; False Positive (FP) means incorrectly detected as an intrusion; True Negative (TN) means correctly detected as benign; False Negative (FN) means incorrectly detected as benign. We use N_* to represent the number of $* \in \{\text{TP, TN, FP, FN}\}$. We compute five evaluation metrics: accuracy $A = \frac{N_{TP}+N_{TN}}{N_{TP}+N_{FP}+N_{TN}+N_{FN}}$, recall $R = \frac{N_{TP}}{N_{TP}+N_{FN}}$, precision $P = \frac{N_{TP}}{N_{TP}+N_{FP}}$, F1 score $F = \frac{2\times P\times R}{P+R}$, and False Positive Rate (FPR) $fpr = \frac{N_{FP}}{N_{FP}+N_{TN}}$. We also provide the receiver operating characteristics (ROC) curve by plotting R against FPR at various threshold settings. The AUC score is defined as the area under the ROC curve.

C. Feature Selection

Here we show the process of our two-step feature selection scheme using NSL-KDD as an example.

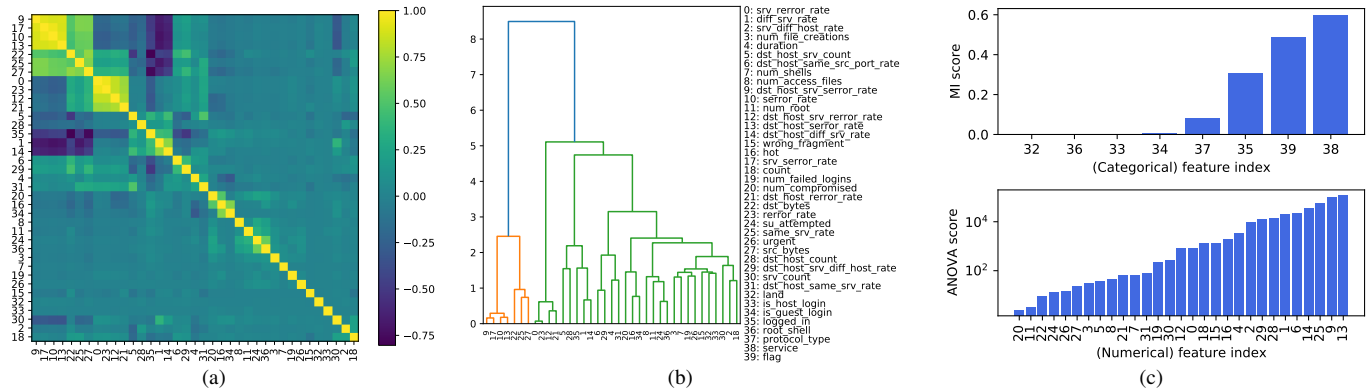
We first remove the zero-variance feature (i.e., 'num_outbound_cmds') from the dataset. Then we evaluate the feature correlation and show the heat map of the correlation matrix in Fig. 4(a). We further perform hierarchy clustering on the computed correlations among features, shown in Fig. 4(b). Focusing on the height at which any two objects are joined together, we can see the height of the link that joins 'feature 0' and 'feature 23' is the smallest, indicating that the two features are the most correlated. We can further obtain the second most correlated feature pair and the third most correlated feature pair. We randomly remove one feature from the three feature pairs as removing these features does not degrade the detection performance.

Second, we evaluate importance scores for all features. We show the ranking of MI scores and ANOVA scores in Fig. 4(c). The MI score of Feature 32, 33, 36 (i.e., 'land', 'root_shell', 'is_host_login') is zero, implying that they are irrelevant to intrusion detection. Therefore, we remove the three features. Note that the ANOVA scores are shown on a logarithmic scale as the scores vary dramatically among different features. Unlike MI scores, ANOVA scores have no zero entries. We start with removing the features with the lowest ANOVA score. We vary the number K of features to remove. Intuitively, useful information may be discarded if K is too large, while redundant information may result in overfitting if K is too small. We tune $K \in \{0, 1, 2, 3, 4\}$ to show the impact of K on detection accuracy. Fig. 5(a) shows the performance of FeCo with different K. We can see that both accuracy and the F1 score increase with $K \leq 3$, implying that removing features with low significance could boost the detection performance. We select $K = 3$ as the accuracy peaks at $K = 3$.

D. Performance of FeCo

We first investigate the performance of FeCo under the centralized setting to focus on evaluating our contrastive-learning-based IDS. We evaluate the impacts of global aggregation of FL in next part. We compare the performance of FeCo to both state-of-the-art IDSs [26], [27], [36] and some other widely-used machine learning baselines including support vector machine (SVM), variational autoencoder (VAE), isolation forest (IsoForest), multilayer perceptron (MLP), logistic regression (LGR), Bernoulli naive Bayes (BNB), K-nearest neighbors (KNN), and decision tree classifier (DTC).

As shown in Sec. IV-C, the threshold for intrusion detection is computed by manually selecting the quantile number p. Fig. 5(b) shows the performance of FeCo with different p value. We can see that the recall increases with p and the FPR also increases as expected. The accuracy and F1 score



(a) (b) (c)

Fig. 4: (a) The heat map of the Spearman correlations. (b) The dendrogram of the correlation clustering. (c) The ranking of MI scores and ANOVA scores for categorical features and numerical features, respectively.

TABLE II: Performance (%) of FeCo in Centralized Setting.

Method	Accuracy	Recall	Precision	F1 score	FPR
FeCo (ours)	**89.55**	**86.80**	94.39	**90.44**	6.82
ESFCM [27]	80.69	80.72	80.85	80.45	-
Two-Tier [36]	-	82.00	-	-	5.43
TDTC [26]	-	84.86	-	-	4.86
VAE	81.77	73.71	92.78	82.15	7.58
IsoForest	79.54	69.35	92.90	79.42	7.00
MLP	79.65	66.41	96.85	78.80	2.85
SVM	77.26	64.65	93.35	76.40	6.09
LGR	75.83	66.05	88.59	75.68	11.24
BNB	77.63	63.27	96.12	76.31	3.38
KNN	77.58	62.39	**97.23**	76.01	**2.35**
DTC	77.36	64.01	94.44	76.30	4.98

first increase dramatically with the increase of p then decrease slowly. Given such results, it is intuitive that one can select the value for p based on the desired FPR. We should select a small p if we desire a low FPR. Ideally, FPR should be very close to the value of $p/100$ if the learned model generalized well on the testing set. In practice, we set $p = 5$, and get $FPR = 6.8\%$ in the testing set. This discrepancy may be due to novel attacks in the testing set. However, we believe that the small gap between $FPR = 6.8\%$ and $5/100$ confirms that FPR would approximate the value of $p/100$.

We show the detection performance of FeCo and other baselines in TABLE II. We set $p = 5$ to obtain these results. FeCo achieves detection accuracy as high as 89.55%. From TABLE II we can see that FeCo outperforms other methods by achieving both the highest recall and the highest accuracy. Note that some entries of TABLE II are missing because they were not provided in the references. To demonstrate FeCo's capability in detecting zero-day attacks, we split the testing attacks into known attacks and novel attacks (i.e., attacks unseen in the training data). We present the recall of FeCo for novel testing attacks in TABLE III. We can see FeCo achieves a recall 8% to 42% higher than other machine learning baselines. SVM, BNB, KNN, DTC, and MLP achieve less than 50% recall, which indicates they are not suitable for detecting zero-attacks.

Among all the methods shown in TABLE II, FeCo, IsoForest, and VAE share a similar detection strategy that detects intrusions by learning a model to characterized benign traffic.

TABLE III: Recall (%) of FeCo for Novel Testing Attacks.

FeCo	VAE	IsoForest	MLP	SVM	LGR	BNB	KNN	DTC
78.45	70.08	61.94	48.43	35.60	52.24	59.84	42.72	49.73

These methods compute a score for inputs using the learned model. The output labels are predicted by comparing the scores to a threshold. Consequently, a different threshold would lead to different detection accuracy. We plot the ROC curves of the three methods in Fig. 5(c). Each point in one curve corresponds to the recall and FPR under one specific threshold. We can see that FeCo achieves the highest recall under the same FPR compare to IsoForest and VAE. Obviously, FeCo outperforms the other two methods in terms of AUC score as well.

To better understand the performance of FeCo, we further look into the recall of FeCo for each category of intrusion attacks (i.e., DoS, Probing, R2L, and U2R). Intuitively, for one attack category, the recall means the proportion of records detected as intrusions, i.e., labelled with '1'.Formally, the recall of each attack category $i \in \{1, 2, 3, 4\}$ (1 for 'DoS', 2 for 'Probe', 3 for 'U2R', and 4 for 'R2L') is defined as

$$R(i) = \frac{\sum_j \mathbb{1}\left(\hat{y}_j = 1 \ \& \ y_j^{multi} = i\right)}{y_j^{multi} = i}, \tag{8}$$

where $\hat{y}_j$ denotes the prediction of FeCo on the j-th data record, and it takes a value of either 0 or 1 as FeCo is a binary IDS. Note that we have true labels of whether a testing record is 'Normal' or 'Intrusion' (i.e., $y_j \in \{0, 1\}$) and whether a testing record is 'Normal', 'DoS', 'Probe', 'U2R', or 'R2L' (i.e., $y_j^{multi} \in \{0, 1, 2, 3, 4\}$).

In TABLE IV, we show the confusion matrix including the attack-specific recall. Here we omit the recall of 'Normal' as normal traffic is not an attack. From TABLE IV, we can see that FeCo achieves higher recall on both DoS attack and Probe attack, while the detection rate on R2L is as low as 0.51. To the best of our knowledge, all works with evaluations on the NSL-KDD dataset show a low detection performance on R2L as well [26], [36]. There are two possible reasons behind the low detection rate on R2L: 1) the number of training instances belonging to R2L attack is as small as 995, and 2) there are

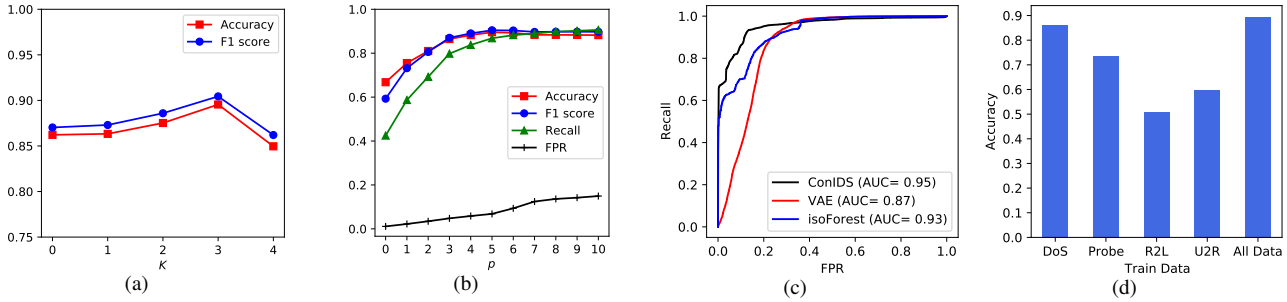


Fig. 5: FeCo performance under the centralized setting. (a) The accuracy and F1 scores of FeCo with different number of removed features K. (b) Accuracy, F1 score, recall, and FPR of FeCo with different percentage value p. (c) ROC curves of FeCo, VAE, and IsoForest. (d) Accuracy of FeCo with training data containing only one attack (DoS, Probe, R2L, or U2R).

many new attacks in the testing set that do not exist in the training set as shown in TABLE I.

To investigate the impact of data distribution, we evaluate FeCo under the scenario when we only have partial data drawn from the whole data distribution. In particular, we sample the data belonging to one attack category (e.g., 'DoS') and the normal data (i.e., 'Normal'). In other words, each FeCo model in these experiments is trained from records of **only one** attack category and 'Normal' class. As a result, the detection performance of such FeCo models is expected to be lower than that of FeCo models trained from the whole NSL-KDD dataset. We evaluate the detection accuracy with the same testing dataset of NSL-KDD and show the results in Fig. 5(d). We can see the detection accuracy with partial data is lower than that with the whole NSL-KDD dataset, which is as expected. The accuracy of the FeCo model trained with only 'R2L' and 'Normal' is as low as 50.8%. This observation indicates that self-learning may result in extremely low detection accuracy as the local data may not exhibit the overall data distribution.

E. Performance of FeCo

Here we focus on the performance of FeCo in the federated setting. To present a thorough comparison, we have three different learning frameworks: FL, self-learning, and centralized learning. In self-learning, the training process is done at the local device using the local data. Centralized learning refers to FeCo investigated in Sec. V-D. Further more, data distribution is one of the biggest factors that impact FL performance. Therefore, we propose to explore three different data distributions, including one IID data distribution and two non-IID data distributions. For the IID data distribution, we randomly select a subset of each class in the whole training dataset as the local data of one client. In non-IID-1, we assume that each client only has $n \in \{1, 2, 3, 4\}$ out of the total four intrusion attack categories. In non-IID-2, each client has $n = 1$ intrusion category. For both non-IID-1 and non-IID-2, we assume all users have benign data, i.e., data from 'Normal' class. non-IID-2 is a special case of non-IID-1 and training on non-IID-2 is more challenging than other settings due to fewer attack categories for training. Meanwhile, we use the same testing data in all settings for fair comparison.

TABLE IV: Detection performance (%) in centralized setting

		Prediction		Total	Recall (%)
		Normal	Intrusion		
Ground Truth	Normal	9049	662	9711	-
	DoS	212	7246	7458	97.16
	Probe	95	2326	2421	96.08
	R2L	1352	1402	2754	50.91
	U2R	35	165	200	82.50

We show the detection performance of FeCo with combinations of the three learning types and three data distributions in TABLE V. In self-learning, the value of evaluation metrics is shown as an averaged value over all clients. Compared with self-learning, FL achieves higher accuracy, recall, and precision regardless of the data distribution. Furthermore, FL achieves similar detection performance under the three data distributions, implying that FeCo is able to obtain stable learning performance with different data distributions. Unlike FL, the performance of self-learning heavily depends on the data distribution. We can see that self-learning achieves as low as 67.66% average accuracy under non-IID-2 data distribution. One noticeable result is that the detection accuracy of FL is still lower than the centralized learning. In practice, centralized learning is difficult to deploy due to privacy concerns. We further show the box-plot of self-learning accuracy of 50 clients in Fig. 6(a) to further accommodate TABLE V (TABLE V only shows the average accuracy among clients). We can see that the variance of the accuracy of non-IID-2 is much larger than those of IID and non-IID-1.

We have another observation when comparing the performance of FeCo with different data distributions but the same learning framework. In the FL setting, the accuracy of FeCo on non-IID-1 is 86.23% which is higher than the 85.65% accuracy achieved on IID data. Similar phenomena occur in self-learning as well. The possible reason behind this is that: the IID data contain a relatively large number of attack sub-classes (i.e., 20) within the small local dataset. Therefore, it becomes difficult to learn a stable representation of benign instances when contrastive learning iteratively pushes benign representations away from so many different attacks. On the contrary, in the non-IID-1 setting, each client possesses only a subset of attack classes thus a smaller number of attack sub-classes. Therefore, it is easier to learn a stable normal

TABLE V: Performance of FeCo in FL, self-learning, and centralized learning frameworks.

Learning Type	Data Distribution	Accuracy	Recall	Precision	F1 score	FPR	DoS	Probe	R2L	U2R
FL	IID	85.65	81.15	92.73	86.55	8.41	92.69	**98.68**	35.00	74.00
	non-IID-1	86.23	82.10	92.89	87.16	8.29	93.12	95.67	41.03	72.50
	non-IID-2	85.53	77.90	**95.93**	85.98	4.26	83.33	95.29	36.49	76.50
Self-learning	IID	81.91	74.53	92.21	82.43	8.33	87.66	92.14	24.33	63.07
	non-IID-1	82.62	75.69	92.39	83.21	8.23	90.48	87.37	25.63	71.81
	non-IID-2	67.66	46.51	93.89	62.21	4.39	54.20	47.22	24.14	59.25
Centralized Learning	-	**89.55**	**86.80**	94.39	**90.44**	6.82	**97.16**	96.08	**50.91**	**82.50**

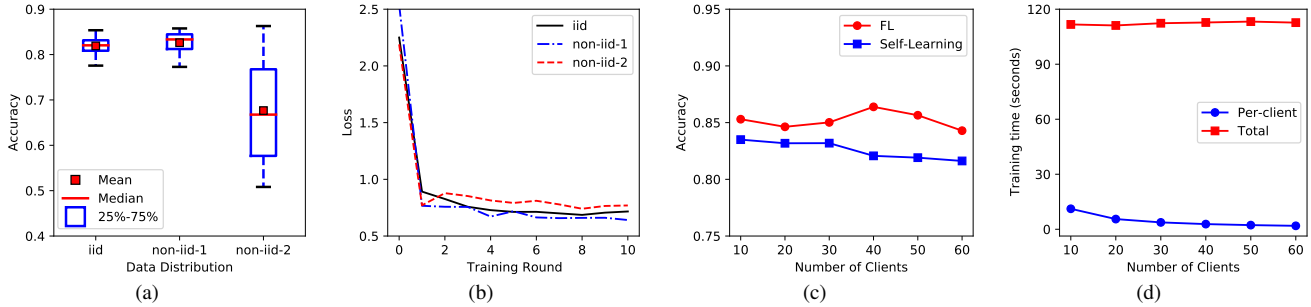


Fig. 6: Evaluations of FeCo. (a) Accuracy distribution of FeCo in self-learning mode. (b) Convergence performance of FeCo in FL mode. (c) Accuracy of FeCo with different number of clients. The size of per-client data get smaller when the number of clients increases since the size of total data is the same; (d) Per-client running time and total running time of all clients.

template. The performance of non-IID-2 is worse than IID possibly because overfitting occurs as the attack categories in the local data are too limited.

We study the convergence performance of FeCo by plotting the value of loss through the training process. As shown in Fig. 6(b), the loss drops sharply at the beginning of the training process, decreases slowly after certain epochs, and finally stays stable. We can see that the loss of FeCo reaches the stable state fast under the three data distributions, implying that FeCo converges after a small number of learning rounds.

We also explore the scalability of FeCo by varying the number of clients. We show the accuracy of FeCo with different number of clients in Fig. 6(c). We can see that the accuracy of self-learning decreases monotonically with the increase of the number of clients. This is because that the size of the local data also decreases with the increase of the number of clients as they are inversely proportional to each other. However, the accuracy of FL does not show a decreasing trend. This observation indicates that FeCo scales well with the number of clients in the FL mode.

We show the overhead of FeCo in Fig. 6(d). Specifically, we present the per-client running time and the total running time of all clients for one FL round. Note that a FL round means that the selected clients finish training their local models and upload the model parameters for one time. We evaluate the running time with the number of clients ranging from 10 to 60. Note that a larger number of clients means a smaller dataset at the local client as discussed above. We can see that the per-client running time decreases with the increase of the number of clients while the total running time stays approximately the same. The per-client running time is as low as 1.878 seconds when the local data contains 2000 records, indicating that FeCo requires relatively little computation resource. We plan to implement FeCo in a gateway device in our future work to demonstrate that FeCo is affordable even in a local gateway with low computation capacity.

VI. Conclusion

In this paper, we propose FeCo, a machine-learning-based IDS for IoT networks. FeCo incorporates contrastive learning into the federated learning framework to support distributed intrusion detection while preserving user data privacy. More importantly, FeCo features a novel detection method based on network traffic representation learning through contrastive learning. While learning for the network traffic representation, FeCo tries to maximize the distance between benign and malicious samples and at the same time minimize the distance among benign samples. The results are that in the new feature space consisting of these learned representations, normal instances lie in a small hyper-sphere. This effectively enables FeCo to achieve better detection accuracy than other baselines as FeCo obtains a more stable normal profile of network traffic. In order to avoid overfitting, we further propose a two-step feature selection scheme to remove redundant features before learning. The feature selection scheme also decreases computation complexity, which makes FeCo more suitable for resource-constrained IoT devices. Through extensive evaluations on the NSL-KDD dataset, we demonstrate the high effectiveness of contrastive learning in IDS with an 8% accuracy improvement over the state-of-the-art. We also investigate FeCo on its convergence performance, overhead, and scalability, demonstrating its applicability for IoT networks.

Acknowledgment

This work was supported in part by the Office of Naval Research under grant N00014-19-1-2621, the National Science Foundation under grants CNS-1837519 and CNS-1916902, and the Virginia Commonwealth Cyber Initiative (CCI).

REFERENCES

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE internet of things journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[2] T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng, "A privacy preserving communication protocol for iot applications in smart homes," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1844–1852, 2017.

[3] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based iot (feciot): Architecture, applications, and research issues," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4118–4149, 2018.

[4] T. D. Nguyen, S. Marchal, M. Miettinen, *et al.*, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, IEEE, 2019.

[5] M. Antonakakis, T. April, M. Bailey, *et al.*, "Understanding the mirai botnet," in *26th {USENIX} security symposium ({USENIX} Security 17)*, pp. 1093–1110, 2017.

[6] D. Oh, D. Kim, and W. W. Ro, "A malicious pattern detection engine for embedded security systems in the internet of things," *Sensors*, vol. 14, no. 12, pp. 24188–24211, 2014.

[7] A. Ferdowsi and W. Saad, "Generative adversarial networks for distributed intrusion detection in the internet of things," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[8] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[9] J. P. Amaral, L. M. Oliveira, J. J. Rodrigues, G. Han, and L. Shu, "Policy and network-based intrusion detection system for ipv6-enabled wireless sensor networks," in *2014 IEEE international conference on communications (ICC)*, pp. 1796–1801, IEEE, 2014.

[10] 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 (AISTATS 17)*, pp. 1273–1282, 2017.

[11] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1, 2021.

[12] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1387–1395, IEEE, 2019.

[13] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6882–6897, 2020.

[14] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[15] V. Jyothsna, R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011.

[16] O. Kopuklu, J. Zheng, H. Xu, and G. Rigoll, "Driver anomaly detection: A dataset and contrastive learning approach," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 91–100, 2021.

[17] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

[18] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 776–794, Springer, 2020.

[19] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?," *arXiv preprint arXiv:2005.10243*, 2020.

[20] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.

[21] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2020.

[22] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, 2021.

[23] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*, pp. 305–316, IEEE, 2010.

[24] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, 2017.

[25] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[26] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi, "Two-tier network anomaly detection model: a machine learning approach," *Journal of Intelligent Information Systems*, vol. 48, no. 1, pp. 61–74, 2017.

[27] S. Rathore and J. H. Park, "Semi-supervised learning based distributed attack detection framework for iot," *Applied Soft Computing*, vol. 72, pp. 79–89, 2018.

[28] N. Wang, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Manda: On adversarial example detection for network intrusion detection system," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10, IEEE, 2021.

[29] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[30] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[31] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[32] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 474–489, IEEE, 2020.

[33] L. Yu, B. Luo, J. Ma, Z. Zhou, and Q. Liu, "You are what you broadcast: Identification of mobile and iot devices from (public) wifi," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 55–72, 2020.

[34] C. Spearman, "The proof and measurement of association between two things.," 1961.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[36] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K.-K. R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 314–323, 2016.

[37] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on abc-afs algorithm for misuse and anomaly detection," *Computer Networks*, vol. 136, pp. 37–50, 2018.

[38] O. Al-Jarrah, A. Siddiqui, M. Elsalamouny, P. D. Yoo, S. Muhaidat, and K. Kim, "Machine-learning-based feature selection techniques for large-scale network intrusion detection," in *2014 IEEE 34th international conference on distributed computing systems workshops (ICDCSW)*, pp. 177–181, IEEE, 2014.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.