Xabier Sáez-de-Cámara, Jose Luis Flores, Cristóbal Arellano, Aitor Urbieta, and Urko Zurutuza. 2023. Federated Explainability for Network Anomaly Characterization. In Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). Association for Computing Machinery, New York, NY, USA, 346–365. https://doi.org/10.1145/3607199.3607234

Federated Explainability for Network Anomaly Characterization

Xabier Sáez-de-Cámara
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA)
Arrasate-Mondragón, Spain
Mondragon Unibertsitatea
Arrasate-Mondragón, Spain
xsaezdecamara@ikerlan.es

Jose Luis Flores
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA)
Arrasate-Mondragón, Spain
jlflores@ikerlan.es

Cristóbal Arellano
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA)
Arrasate-Mondragón, Spain
carellano@ikerlan.es

Aitor Urbieta
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA)
Arrasate-Mondragón, Spain
aurbieta@ikerlan.es

Urko Zurutuza
Mondragon Unibertsitatea, Faculty of Engineering, Electronics and Computing
Arrasate-Mondragón, Spain
uzurutuza@mondragon.edu

ABSTRACT

Machine learning (ML) based systems have shown promising results for intrusion detection due to their ability to learn complex patterns. In particular, unsupervised anomaly detection approaches offer practical advantages as does not require labeling the training data, which is costly and time-consuming. To further address practical concerns, there is a rising interest in adopting federated learning (FL) techniques as a recent ML model training paradigm for distributed settings (e.g., IoT), thereby addressing challenges such as data privacy, availability and communication cost concerns. However, output generated by unsupervised models provide limited contextual information to security analysts at SOCs, as they usually lack the means to know why a sample was classified as anomalous or cannot distinguish between different types of anomalies, difficulting the extraction of actionable information and correlation with other indicators. Moreover, ML explainability methods have received little attention in FL settings and present additional challenges due to the distributed nature and data locality requirements. This paper proposes a new methodology to characterize and explain the anomalies detected by unsupervised ML-based intrusion detection models in FL settings. We adapt and develop explainability, clustering and cluster validation algorithms to FL settings to mine patterns in the anomalous samples and identify different threats throughout the entire network, demonstrating the results on two network intrusion detection datasets containing real IoT malware, namely Gafgyt and Mirai, and various attack traces. The learned clustering results can be used to classify emerging anomalies, provide additional context that can be leveraged to gain more insight and enable the correlation of the anomalies with alerts triggered by other security solutions.

RAID 2023, October 16–18, 2023, Hong Kong, China

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; *Distributed systems security*; • **Computing methodologies** → *Cooperation and coordination*; *Unsupervised learning*.

KEYWORDS

explainable artificial intelligence, federated learning, IoT security, SIEM

ACM Reference Format:
Xabier Sáez-de-Cámara, Jose Luis Flores, Cristóbal Arellano, Aitor Urbieta, and Urko Zurutuza. 2023. Federated Explainability for Network Anomaly Characterization. In *Proceedings of The 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2023).* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The use of an Intrusion Detection System (IDS) is a common practice for monitoring assets both in IT [1] and Internet of Things (IoT) [2] infrastructures. Over the last decades, there has been an increasing interest in machine learning (ML) and deep learning (DL) algorithms for the development of IDS, as they can show greater generalization capabilities over traditional rule-based systems [3–5].

ML and DL-based IDS generally take three different approaches, among others, for data modeling: supervised [6], unsupervised [7] and semi-supervised [8]. Training supervised learning models require the data to be labeled into a finite set of classes, each representing a specific malicious activity and a class for normal (benign) activity. The objective is then to classify new incoming data samples into those classes. Regarding unsupervised approaches, they are popular for anomaly detection, where the model learns a representation of the legitimate or benign behavior and flags all samples that deviate from that baseline as anomalous. Unsupervised approaches do not need labeled data, but they typically assume that the training samples are benign. Lastly, semi-supervised models follow a hybrid

approach where labeling information is limited to a subset of the training data while the rest are unlabeled.

Regarding IDS model training paradigms, a current direction gaining substantial interest is the adoption of federated learning (FL) [9]. FL is a distributed setting to train a single global model between multiple collaborating clients. The training data of each client is kept local at each device and never shared; instead, they partially train a model and communicate the trained parameters to a central server. The central server aggregates the model updates provided by all the clients to create the global model. This training process is usually repeated iteratively until the global model converges. FL provides greater data privacy and network communication efficiency, which can be preferable to traditional cloud or edge training settings.

However, despite the promising results of ML and DL for intrusion detection, multiple issues still limit the practical and widespread adoption of these methods. Challenges such as the availability and cost of data labeling or the need for more interpretation and explainability of the results have been [10] and continue to be [8] present since the use of these methods for intrusion detection.

Regarding the interpretation of the results, an advantage of supervised approaches is that by predicting the label of a malicious event, the security analysts have an idea of what kind of threats they face, assuming a correct classification. However, obtaining labeled data for model training is costly, time-consuming, and often infeasible in practical scenarios. Additionally, using a finite set of classes limits the types of attacks that a supervised model can detect. In contrast, anomaly detection systems with unsupervised approaches do not need labeled data and can potentially find novel attacks; however, the ability to characterize the detected anomalies to automatically provide an interpretation gets limited.

Recently, the ML community has increased its efforts in the field of eXplainable Artificial Intelligence (XAI) to provide interpretation and explainability of both the models and the predictions made with them [11], and thus, address the black box nature of ML and especially DL models. Some IDSs have also adopted XAI methods, which is crucial to increase the trust of these techniques by security analysts [12].

However, the integration of XAI methods into FL is an area that has received little attention and presents additional challenges due to the particularities of this setting. Implying that while IDS training can benefit from FL properties, such as a bigger pool of training data, higher communication efficiency or increased data privacy (although attacks against FL and defenses exist [13]), SOC analysts lack an increased trust or visibility offered by XAI methods as they are underdeveloped in this settings. For instance, FL challenges, including the distributed nature of the datasets, high heterogeneity regarding data distribution and client capabilities, large scale in terms of the number of clients in the federated network, and the need to maintain the training data local to each client are issues that need to be considered for using XAI methods into FL [14, 15]. In such a distributed and heterogeneous network, raised anomalies can greatly vary between clients, difficulting the overall visibility of the security in the network. Providing a common ground to ensure the same anomalous events happening in different clients are explained in the same terms by means of a federated XAI method, it could allow analysts to better assess the security level of the network

This work aims to fill this gap by proposing a method to characterize and explain the anomalies detected by unsupervised ML/DL-based IDS models in a FL setting. In particular, we use XAI and clustering techniques to explain anomalies and group common anomalous patterns. The method is evaluated on anomalies generated by network attacks from real IoT malware, namely Gafgyt and Mirai [16]. Each client in a FL setting might be exposed to different attacks; hence, by characterizing the anomalies in a federated way, all clients can be aware of the various anomalous patterns that have occurred across the federated network. In summary, our work aims to address the following questions in the context of FL:

(1) What features have been the most decisive in classifying those samples as anomalous?
(2) Can the explainability model identify different groups of anomalies?
(3) What do all anomalies in a specific group have in common?

The contributions can be summarized as follows:

- We introduce a novel methodology to explain and characterize anomalies generated by ML/DL-based anomaly detection models in a FL setting. Particularly, the characterization is based on training SHAP [11] explainability models in a federated way. Additionally, to make all the clients aware of the various anomalous patterns that occurred across the whole network, we leverage a federated version of k-means [17] and also adapt a clustering internal validation metric to be computed in a distributed manner.
- We perform an experimental validation of the methodology on two different IoT network security datasets with a wide variety of attacks and malicious behaviors. The first is based on network packet-level data from the Gotham Testbed [18], and the second uses network flow-level data from N-BaIoT [19]. Autoencoders are used as the anomaly detection model for both datasets.
- We show the results of the generated explanations and the characterization of the anomalies. Additionally, we leverage IDMEF as an alert message exchange format to enable the interoperability of the proposed method with third-party security solutions such as SIEMs so that the characterized anomalies can be used for correlation with events generated by other data sources.

The source code for the implementation is available at [20].

2 RELATED WORK

Many applications require both high accuracy and interpretability of the results. For the latter, various methods have been proposed to help interpret the predictions of complex models. Lundberg et al. present SHAP (SHapley Additive exPlanation) [11], a framework for interpreting predictions that unifies six existing explainability methods, including LIME, DeepLIFT and classical Shapley values. The process requires a certain number of training data samples as a baseline for the computation of the SHAP values. For an explanation method designed for models applied to general cybersecurity applications, Guo et al. present LEMNA [21], an explainer based on a mixture regression model and fused lasso penalty term. They test the method in malware classification for PDF files and binary reverse-engineering examples.

2.1 Explainability for cybersecurity anomaly or attack detection in non-FL settings

Most of the works regarding XAI techniques for cybersecurity are mainly focused on using them as an end for visualization or model/prediction verification purposes.

Wang et al. [22] propose a framework that uses SHAP to provide local and global explanations of IDS to help security analysts interpret the predictions. Explanations are evaluated and compared for two supervised models trained on the NSL-KDD dataset[1]. They show how different attack types generate different SHAP value patterns; however, they only use it for visualization purposes and do not discuss analysis on top of these values to extract further information.

Antwarg et al. [23] use SHAP to explain anomalies detected by an unsupervised autoencoder model to provide additional information for domain experts. They first identify the features with high reconstruction error and then use SHAP to explain them. They evaluate the proposal on the KDD Cup 1999 dataset, among other datasets from different fields. The explanations are visualized for easier understanding and triaging of anomalies.

Liu et al. present FAIXID [24], a generic framework to add explainability to IDS at different layers. The layers include data cleaning, explaining the internals of a trained supervised model, local explanations of the predictions, and presenting the results to security analysts using different visualizations depending on the expertise or role of each analyst.

Rao et al. [25] train an isolation forest on the NSL-KDD dataset to classify normal and anomalous samples. They use SHAP and LIME to extract and visualize explanations. In addition, they autogenerate labels for the attacks by assigning to each anomaly the name of the most important feature to make the prediction.

Other proposals leverage or provide additional analysis on top of the explainability results to extract further information from the detected anomalies or predicted classes.

Nguyen et al. present GEE [26], an explainable variational autoencoder (VAE) for network anomaly detection, which is tested on NetFlow traces from the UGR16 dataset. In addition, they provide a gradient-based technique to explain the anomalous samples by identifying the main features that cause the anomaly. Furthermore, they use gradient information as a fingerprint to group similar anomalies. However, this particular point is underexplored in the paper, and the gradient method is specific to the VAE model. Liyanage et al. [27] leverage GEE to develop a framework for characterizing attacks from network flow anomalies. Instead of using XAI techniques or GEE's gradient-based explanation methods, they use two levels of frequent itemset mining (FIM) to extract anomalous data patterns. Some steps of the mining require labeled data samples.

Barnard et al. [28] present a network IDS divided into two stages. The first stage involves training a supervised XGBoost model for binary classification of network flow data and SHAP to explain the predictions. The second stage trains an autoencoder which uses as input the SHAP explanations from the previous stage. The central hypothesis they are testing is whether the system can use the first stage to distinguish normal from anomalous flows, and the second stage to distinguish known from unknown behavior. The proposal is evaluated on the NSL-KDD dataset. However, the second stage is tightly coupled to the first one, which requires labeled data, and they do not consider the characterization of different attack behavior clusters within the explanations.

Sudheera et al. [29] develop ADEPT, a framework for network flow anomaly detection and attack-stage identification in a distributed IoT network based on multiple clients and a centralized server. It works in three phases. Each client locally detects anomalous network flows, which are then sent to the central server. Then, the central server uses FIM for data mining across all the anomalous flows from all the clients. While explainability is not regarded in this work, the patterns extracted with FIM have the benefit of being interpretable. Finally, the malicious flows are classified into attack stages using supervised learning approaches, which require ground truth data labels. While their distributed approach benefits from improved privacy and reduced bandwidth compared to a fully centralized one, anomalous flows containing sensitive data are still sent to the central server. In contrast, FL architectures can offer greater privacy and data reduction while still being able to cooperate with clients.

2.2 Explainability for cybersecurity in federated learning settings

Haffar et al. [30] use random forests (RF) as surrogates of the supervised FL model. Each client in the network trains a RF using its local training data. When the FL model misclassifies a sample, they leverage the trees in the RF to compute feature importance values. The feature importance is used to detect and explain attacks against the FL model training process. The explanations are performed at the client level and require labeled training data. Each client has its own explainer model, which might differ from the rest as they are trained independently and not in a federated way, difficulting the interpretation of the explanations globally. Their focus is not on explaining and characterizing predictions but on detecting potential attacks against the FL training process.

Huong et al. [31] propose a FL-based anomaly detection architecture for industrial control systems. They use SHAP to interpret and verify the trained FL model, and provide visualizations as a supporting tool to domain experts. The SHAP model explainer itself is not trained in a federated way. SHAP needs background data samples as a baseline; however, the authors do not discuss how the baseline is extracted, which should be given special consideration due to the distributed nature of data in FL settings.

2.3 Discussion

Most of the literature regarding XAI techniques is focused on using explanations for visualization and model verification purposes. Meanwhile, works that leverage and build on top of the explanations to provide additional functionalities (such as giving context to anomalies in order to group or characterize them) are scarce and are designed for centralized or distributed architectures that do not offer the same benefits as FL. Moreover, most works require labeled data in certain stages of their proposal [22, 24, 27–30], which might not be feasible in practical settings.

[1] https://www.unb.ca/cic/datasets/nsl.html

Additionally, while some works use XAI techniques in FL settings, the objective is to verify the FL training process to detect adversarial attacks [30] or visualize and verify the trained model [31]. The explainer models were not trained in a federated way; this requires breaking the FL assumptions/properties to offer the explanations or using a different explainer on each client, difficulting the interpretation across the federated network because the same sample can have different explanations on different clients.

None of the works using SHAP in FL consider or discuss how to extract a baseline for SHAP in a federated way, which is required to generate the explainer. The baseline selection is critical in SHAP because the generated explanations depend on them [32, 33]. In FL, explanations from all the clients should have a "common ground" so the same event happening in different clients is explained in the same terms so the information can be shared with all the federated devices.

To the best of our knowledge, this work is the first to leverage the explanations generated with XAI techniques in a FL architecture in order to give context to the alerts produced by unsupervised ML-based IDS models for clustering, characterization and autolabeling of the anomalous events. All steps are performed in an unsupervised way.

3 PROPOSED SYSTEM MODEL

In this section, we first describe the considered FL setting and the threat model. Then, we provide background knowledge on the SHAP explanation model. Finally, we present an overview of the proposed method's architecture.

3.1 Federated learning setting

The proposed system is designed to be deployed in a standard cross-device FL architecture [34], composed of many clients and a single FL aggregation server. Low-powered IoT clients are expected to be connected to the FL aggregation server via a hub or gateway. Meanwhile, more capable IoT clients or other endpoints might be directly connected to the aggregation server without any intermediary.

All data (training and evaluation data) is generated locally at each device and remains decentralized throughout the process, including at the model training and explanation generation phases. The aggregation server coordinates the process and only receives model updates or highly aggregated data.

All the clients are expected to be able to perform ML model training and inference tasks. We do not assume any particular ML model for the unsupervised anomaly detection process. The explanations are also performed independently of the selected ML model, as we are adopting the model-agnostic Kernel SHAP algorithm to generate the explanations (further detailed in section 3.3). However, to perform the implementation and evaluation (section 5.2), we use autoencoders due to the successful application of these models for unsupervised network anomaly detection [7, 19].

3.2 Threat model

We consider a common IoT threat model consisting of different attack stages which generate various network patterns. These stages are modeled based on the Mirai [16] malware behavior, including: (i) network scanning to find vulnerable devices, (ii) brute-forcing attacks, (iii) periodic communication with the command and control (C&C) server, (iv) ingress tool transfer or malware downloading into the compromised devices, (v) remote command execution and (vi) denial of service (DoS) attacks. The different malicious activities are distributed across time and space, not all devices will be exposed to the same attacks, and they may occur at different times.

This work assumes that no IoT device is compromised prior to the FL model training. However, they can be attacked or compromised after model training and during the generation of explanation models. We also assume that no device behaves in an adversarial manner. Model poisoning attacks against FL [35] are outside the scope of this paper.

3.3 SHAP background

Lundberg et al. [11] introduce the observation that any explanation for the prediction of a model f is itself a model g. Here, g is the explanation model, a simpler and more interpretable model that approximates f. They focus on explanation models following additive feature attribution methods, a linear function of binary variables. The binary variables are simplified inputs x', where $x' \in \{0,1\}^M$ and M is the number of features. The simplified inputs are derived from the original inputs x by a mapping function $x = h_x(x')$ defined for each input. Additive feature attribution methods follow the definition shown in equation (1).

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{1}$$

The $\phi_i \in \mathbb{R}$ values represent the importance effect of the corresponding feature for a particular prediction. The class of additive feature attribution methods presents a unique solution where the ϕ_i values are the Shapley values [11] from cooperative game theory. The computation of Shapley values involves testing different subsets of data features, and the importance value is assigned based on the effect on the model prediction of including that feature. SHAP values are adapted Shapley values; since most ML models cannot handle changes in the number of sample features, SHAP represents an absent feature by approximating it using a conditional expectation function of f. The SHAP value of a particular feature gives the change in the expected model prediction with respect to the base value when conditioning on that feature. Adding the SHAP values of all the features results in the same value as the prediction $f(x)$. The base value (ϕ_0) represents the value that would be predicted if all the sample features were absent.

Kernel SHAP is a model-agnostic method to approximate those values more efficiently than classic Shapley sampling methods. The method requires a background dataset to compute the expected values. For large datasets, this background data is usually subsampled from the training data because the computation time for the SHAP values increases linearly with the size of the background data. However, in FL settings, the dataset is distributed across all the clients, and no party (including the FL aggregation server) can directly access the raw data of others. Therefore, the selection of the background data requires special attention in order to capture representative samples from all the clients in the network while also respecting data locality requirements (privacy reasons) and data

transmission volume minimization. Ensuring that all the clients use the same background samples will guarantee that all the explanations provided by the clients are computed with respect to the same background and be comparable to each other.

3.4 Architecture of the proposed method

The diagram of all the components involved in the proposed method is shown in Figure 1. The diagram is divided into three main blocks: (i) anomaly detection model training, (ii) model inference and (iii) explainer model training and the characterization of the anomalies.

The main focus of this manuscript is not on the FL anomaly detection model training or inference, but on the third block regarding the FL explainer training and anomaly characterization, as denoted by the steps with a shaded background in Figure 1.

As shown in Figure 1, the last block includes two steps that are performed in a federated way: the explainer model training and the characterization of the anomalies. We will use Kernel SHAP to train the explainer model, and as mentioned in section 3.3, it requires two inputs, the prediction model f and a background dataset. The output of this step is the explainer model g. The prediction model f is the global anomaly detection model trained with FL, which is common to all clients. To ensure that all clients have the same explainer model g, the same background dataset must be used, which is usually a representative subsample of the training data. However, the data in FL settings are distributed across all clients and not shared. To generate a common representative background set as a subsample of the entire distributed dataset, we will leverage and adapt a federated version of k-means based on k-FED [17]. In this step, the k from k-means refers to the number of subsampled data samples to be used as the background for SHAP.

The anomaly characterization process is the second step that requires the use of FL. One of the inputs of this process is the generated explanations for the anomalous samples, that is, the ϕ_i SHAP values showing the importance of each feature. The other inputs are the processed data and the raw data of the anomalous samples. While some features, such as source and destination IP addresses or timestamps, are not suitable as inputs to the ML model to prevent learning from spurious correlations [36, 37], they are certainly valuable for security analysts for correlating with other events. Hence, we use both for the characterization. Since the anomaly explanations are local to each client, we use FL to ensure that all clients are able to know and identify all the different anomalous activities found across the federated network, even if each client has been exposed to a different set of attacks. Specifically, we will again leverage k-FED [17] to group the explainability results in each client and share it with other peers in the network so that all can have the same clustering labels to refer to the same anomalous instances. In this step, k refers to the global number of anomalous behaviors found throughout the federated network.

4 ALGORITHM DETAILS

In this section, we detail the procedures to perform the explainer model training and the anomaly clustering in a FL setting. Additionally, we describe the cluster explanation and the anomaly message exchange format.

Algorithm 1: FL training for the Kernel SHAP explainer model.

Input: A set of clients Z each with local data $\boldsymbol{N}^{(z)}$, local number of clusters for each client $k^{(z)}$ and number of global clusters k.
Result: A trained Kernel SHAP model at each client.

foreach *client* $z \in Z$ **in parallel do**
  Run k-means with $k^{(z)}$ in local data $\boldsymbol{N}^{(z)}$ and obtain cluster centers $\Theta^{(z)} = (\boldsymbol{\theta}_1^{(z)}, \ldots, \boldsymbol{\theta}_{k(z)}^{(z)})$.
  Compute number of data samples in each cluster $C^{(z)} = (c_1^{(z)}, \ldots, c_{k(z)}^{(z)})$.
  for $i \in \{1, 2, \ldots, k^{(z)}\}$ **do**
    for $j \in \{1, 2, \ldots, d\}$ **do**
      $\mathrm{t} \leftarrow \mathrm{argmin}_{\mathrm{t}}(|\boldsymbol{N}^{(z)}[\mathrm{t}, j] - \boldsymbol{\theta}_i^{(z)}[j]|)$
      $\boldsymbol{\theta}_i^{(z)}[j] \leftarrow \boldsymbol{N}^{(z)}[t, j]$.
    end
  end
  Send $\Theta^{(z)}$ and $C^{(z)}$ to the central server.
end
Pick any $z \in [Z]$ and let $M \leftarrow \Theta^{(z)}$ (in server).
while *there are less than* k *points in* M **do**
  Let $\bar{\boldsymbol{\theta}} \leftarrow \mathrm{argmax}_{z \in [Z], i \in [k]} d_M(\boldsymbol{\theta}_i^{(z)})$. That is, the farthest $\boldsymbol{\theta}_i^{(z)}$ from the set M.
  $M \leftarrow M \cup \{\bar{\boldsymbol{\theta}}\}$.
end
Run one round of Lloyd's heuristic (k-means), using the points in M as initial centers, to cluster points $\boldsymbol{\theta}_i^{(z)}$, $z \in [Z]$, $i \in [k]$ into k clusters: $\mathrm{B} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k)$.
$C \leftarrow$ Estimate total number of data samples in the global clustering results using B and $C^{(z)}$.
for $i \in \{1, 2, \ldots, k\}$ **do**
  $\boldsymbol{b_i} \leftarrow \Theta^{(z)}[\mathrm{argmin}(d(\boldsymbol{b_i}, \Theta^{(z)}))]$.
end
foreach *client* $z \in Z$ **in parallel do**
  Receive B and C from the server.
  Train Kernel SHAP model (B, C).
end

As explained in section 3.4, we leverage and adapt k-FED [17] for both FL processes. k-FED includes several practical advantages that make it suitable for large federated networks. First, it is a one-shot process that only requires a single round of communication to compute the global clustering results, significantly reducing the communication overhead. Second, the computation is done locally at each client and is independent of each other; therefore, it does not require synchronization and can be easily parallelized.

4.1 Federated learning for explainer model training

The FL method to train the Kernel SHAP explainer models is described in Algorithm 1. The main objective of this algorithm is to compute SHAP background baseline samples common to all the clients in the federated network. Using a notation similar to the one in k-FED [17], $\boldsymbol{N}^{(z)} \in \mathbb{R}^{n^{(z)} \times d}$ denotes the local training dataset of

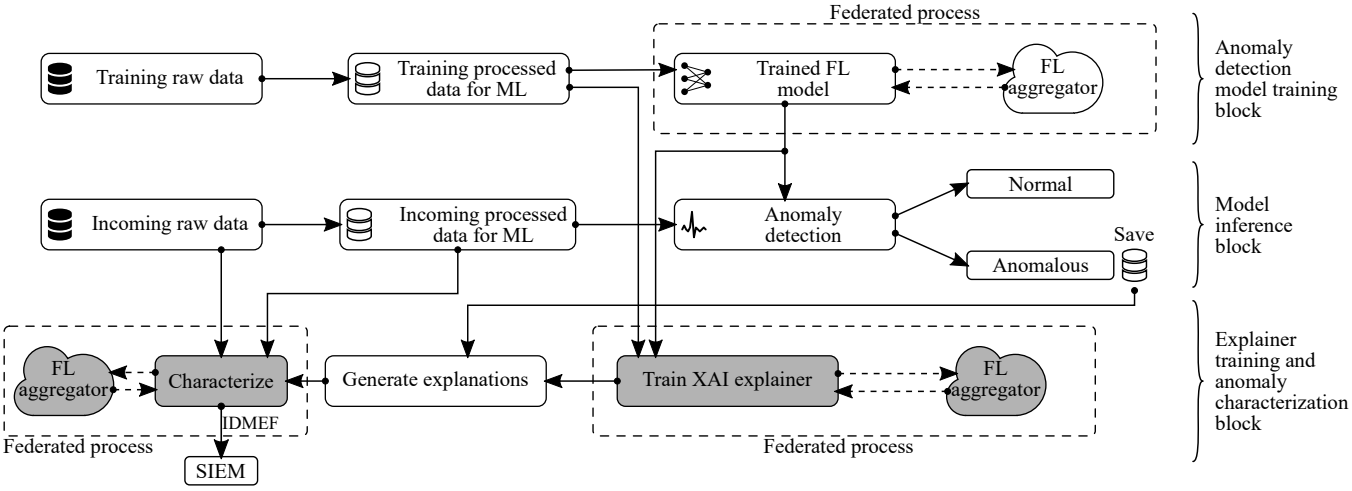


Figure 1: Diagram of the proposed methodology. The components within the dashed frames represent steps performed using FL. Components with shaded background refer to the contributions of this paper.

a particular client z with $n^{(z)}$ local training samples, each having d dimensions.

Each client computes a local k-means process (as described in the first algorithm from [17]) using $k^{(z)}$ centroids. Unlike the original k-FED (described in the second algorithm from the same reference), we include two additional steps to adapt it for SHAP background data extraction. First, we compute the number of data samples in each cluster ($C^{(z)}$, line 3 in Algorithm 1). Then, we round the obtained cluster center values so that the features of each center are equal to the value of the closest feature in $\boldsymbol{N}^{(z)}$ (lines 4–9). This rounding step is included to match the non-federated implementation of k-means sampling in the SHAP source code [38]. The rounded cluster centers $\Theta^{(z)}$ and the $C^{(z)}$ are sent to the server.

At the server, the global clustering into k groups is performed in the same way as in k-FED. However, we again include two additional postprocessing steps. We estimate the total number of samples in each global cluster (C, line 18) based on the received $C^{(z)}$ values and the final clustering result B. Then, each global centroid is assigned to the nearest center from the local cluster candidates $\Theta^{(z)}$ (lines 19-21). This rounding step is performed to ensure that the final clustering centroids include values representative of the training data from all $\boldsymbol{N}^{(z)}$.

The global results B and C are sent to all clients; therefore, they use the same values as SHAP background baseline samples during the Kernel SHAP model training.

4.2 Federated learning for anomaly clustering

The anomaly clustering process across the federated network is detailed in Algorithm 2. The main objective of this step is to compute a global clustering of the anomalies found across all the clients using FL. The results are shared with all clients, so they can identify and know all the found activities throughout the network (even if each client has not been exposed to all attacks or no attack at all).

Using the explainer model trained with Algorithm 1, first, each client computes the SHAP values of all its anomalous samples

Algorithm 2: FL clustering of the SHAP values of the identified anomalies.

Input: A set of clients Z with anomalous samples, and number of global clusters k.
Result: The global clustering results of the anomalies across all the clients in the federated network.

1 **foreach** *client* $z \in Z$ **in parallel do**
2 Run trained Kernel SHAP on the anomalous samples to create the local dataset $\boldsymbol{\Phi}^{(z)}$.
3 $\boldsymbol{\Phi}^{(z)}_{\text{norm}} \leftarrow$ independently scale samples from $\boldsymbol{\Phi}^{(z)}$ to unit norm.
4 Estimate $k^{(z)}$ with HDBSCAN on $\boldsymbol{\Phi}^{(z)}_{\text{norm}}$. Send it to the server.
5 **end**
6 $k' \leftarrow \max_z(k^{(z)})$. Send k' to the clients.
7 **foreach** *client* $z \in Z$ **in parallel do**
8 Run k-means with k' in local data $\boldsymbol{\Phi}^{(z)}_{\text{norm}}$ and obtain cluster centers $\Sigma^{(z)} = (\boldsymbol{\sigma}^{(z)}_1, \ldots, \boldsymbol{\sigma}^{(z)}_{k'})$.
9 Send $\Sigma^{(z)}$ to the central server.
10 **end**
11 Pick any $z \in [Z]$ and let $M \leftarrow \Sigma^{(z)}$ (in server).
12 **while** *there are less than k points in M* **do**
13 Let $\bar{\boldsymbol{\sigma}} \leftarrow \text{argmax}_{z\in[Z],i\in[k]} d_M(\boldsymbol{\sigma}^{(z)}_i)$. That is, the farthest $\boldsymbol{\sigma}^{(z)}_i$ from the set M.
14 $M \leftarrow M \cup \{\bar{\boldsymbol{\sigma}}\}$.
15 **end**
16 Run one round of Lloyd's heuristic (k-means), using the points in M as initial centers, to cluster points $\boldsymbol{\sigma}^{(z)}_i$, $z \in [Z]$, $i \in [k]$ into k clusters: S = $(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_k)$. Send S to clients.

($\boldsymbol{\Phi}^{(z)} \in \mathbb{R}^{n^{(z)}_{\text{anom}} \times d}$). Then, each sample from $\boldsymbol{\Phi}^{(z)}$ is scaled to unit norm to create $\boldsymbol{\Phi}^{(z)}_{\text{norm}}$; this step is performed so that the subsequent clustering steps give more weight to the direction of the SHAP values instead of the magnitude.

After the normalization step, each client locally applies the HDBSCAN clustering algorithm to automatically estimate the number

of clusters ($k^{(z)}$) in $\Phi^{(z)}_{\text{norm}}$ (at this step, we are only interested in the local estimation of the number of clusters, not the clustering results themselves). HDBSCAN is used due to easier and more intuitive hyperparameter selection compared to sweeping through different values of $k^{(z)}$, clustering the data, and then using internal clustering validation metrics to evaluate the fitness, which might require manual inspection to interpret the fitness results. Since the number of clients in a FL setting can be very large, using HDBSCAN can improve the automation of this process.

After estimating $k^{(z)}$ in each client, the value is sent to the server. The server selects k' as the maximum $k^{(z)}$ for all clients z. k' is the number of clusters per device, and k is the total number of clusters over the federated network.

The rest of the federated k-means clustering is performed in the same way as in k-FED. In the end, all clients will have the clustering results S corresponding to the different anomalous patterns found throughout the network.

In Algorithm 2, we assume for simplicity that k is known and it is an input of the algorithm. However k will be unknown in practice, as it refers to the number of anomalous behavior clusters found throughout the network. Therefore, to address this issue, we are going to consider k as unknown and will estimate and select it based on unsupervised internal clustering validation metrics, with the added complexity that the metric must be computed efficiently in a federated (distributed) setting. For this purpose, we are going to adapt the Calinski-Harabasz (CH) score to a FL setting, shown in Algorithm 3. Since the CH score is based on the between-group and within-group sum of squares ratios, these values can be easily computed in a distributed setting and only incur minimal transmission costs. Alternative metrics, such as the Silhouette score, may not be suitable in FL settings because it requires computing pairwise distances between all the samples. Since samples in the same cluster can be distributed among different clients, this would require higher data transmission and computation costs.

For the estimation of k, we repeat the steps from lines 7–16 in Algorithm 2 for different values of k, starting from k' to no more than $k'|Z|$. After each repetition, we use Algorithm 3 to measure the clustering performance, where a higher CH score indicates a better fit.

Algorithm 3: Computation of the Calinski-Harabasz score in a federated (distributed) way.

Input: A set of clients Z with local data $\mathrm{X}^{(z)}$, cluster labeling results $\mathrm{L}^{(z)}$ for the local data and global cluster centers GC.
Result: Calinski-Harabasz score CH.

1 $K \leftarrow$ total number of unique labels (clusters).
2 **foreach** *client* $z \in Z$ **in parallel do**
3 $\quad n^{(z)} \leftarrow$ number of samples in $\mathrm{X}^{(z)}$.
4 $\quad s^{(z)} \leftarrow$ sum of $\mathrm{X}^{(z)}$ along the columns (features).
5 $\quad$ Send $n^{(z)}$ and $s^{(z)}$ to the server.
6 **end**
7 $N \leftarrow$ sum of $n^{(z)}$ for all clients $z \in Z$. (total number of samples)
8 $C_g \leftarrow \frac{\text{sum of } s^{(z)} \text{ along columns for all clients } z \in Z}{N}$. (dataset center)
9 $WGSS \leftarrow 0$ (within-group sum of squares).
10 $BGSS \leftarrow 0$ (between-group sum of squares).
11 **for** k *in range* K **do**
12 $\quad$ **foreach** *client* $z \in Z$ **in parallel do**
13 $\quad\quad \mathrm{X}^{(z)}_k \leftarrow \mathrm{X}^{(z)}$ where $\mathrm{L}^{(z)} = k$.
14 $\quad\quad W^{(z)} \leftarrow$ sum of squared distances between $\mathrm{X}^{(z)}_k$ and GC_k.
15 $\quad\quad n^{(z)}_k \leftarrow$ number of samples in $\mathrm{X}^{(z)}_k$.
16 $\quad\quad$ Send $W^{(z)}$ and $n^{(z)}_k$ to the server.
17 $\quad$ **end**
18 $\quad WGSS \leftarrow WGSS+$ sum $W^{(z)}$ for all clients $z \in Z$.
19 $\quad BGSS \leftarrow BGSS+$ (sum $n^{(z)}_k$ for all clients $z \in Z$) $\times$ squared distance between GC_k and C_g.
20 **end**
21 $\mathrm{CH} \leftarrow \frac{BGSS}{WGSS} \frac{N-K}{K-1}$.

4.3 Explaining clusters

After executing the steps described in Algorithms 1 and 2, each client has the information about which features have been the most decisive in classifying the samples as anomalous by means of the SHAP values. Additionally, those samples can be grouped using the global clustering results computed using FL. Thus, groups of anomalies can be broadly characterized by the SHAP values of their corresponding cluster center.

However, SHAP values only give the importance of a feature for the prediction of the model, not the actual values of said feature. To find out which feature values the anomalies for a specific cluster have in common, we will compute basic summary statistics (e.g., min, max, mean, std, percentiles) over the features of all anomalous samples for each cluster. More sophisticated data extraction processes could be used to extract additional patterns from the anomalies in each group. While a detailed description of those methods is considered outside the scope of this paper, some examples could include frequent itemset mining to extract the repeating feature patterns within a cluster, or training shallow decision trees or linear models to identify the most salient feature ranges that differentiate one cluster of anomalies from the rest of clusters in a one-vs-all approach or considering all clusters simultaneously.

4.4 Anomaly message exchange

Grouping the anomalies based on a similar explanation allows several benefits, such as capturing more context of the events, alert volume reduction (reducing overload for the security analysts) and fewer data transmission costs when exchanging the alerts to a security management system (e.g., SIEM). Anomalies detected in the new incoming data can be automatically assigned to one of the learned global clusters based on proximity (in the explanation space) to the nearest center and auto-tagged with the cluster's index to provide a context that is common to all the clients in the federated network.

Additionally, since a single attack can generate multiple anomalous activities corresponding to different clusters, providing alert messages related to the temporal correlation of the anomalous clusters occurring simultaneously could help identify the tools or methods used to perform the attack.

To this end, using a standard anomaly message exchange format makes communicating with all the clients in the network easier. Moreover, it allows interoperability with other intrusion detection

systems and event correlation engines to create actionable information by combining alerts from this type of unsupervised anomaly detection systems and other traditional solutions.

To allow this interconnection and communication, we rely on the Intrusion Detection Message Exchange Format (IDMEF) described in RFC 4765 and RFC 4766 [39]. The `Alert` IDMEF message type provides a way to describe detailed event information. In our case, we use the `Alert` type to create a message generated by a group of anomalous events corresponding to a single cluster. To also provide temporal correlation of anomalies falling into different clusters at similar time windows, we use the `CorrelationAlert` class, which groups one or more `Alert` messages.

Since we still keep the raw network data available for the anomaly characterization step, as shown in Figure 1, we can populate the `Source` and `Target` classes with information regarding the source and destination addresses involved in the anomalous events. As the `Classification` class, we include the cluster's index of the anomalies. Besides, the `AdditionalData` class allows us to include the relevant context regarding the group of anomalies, such as the summary statistics described in the previous section. Since the average SHAP values of each cluster (centroids) are known to all clients and the FL server, they can also be sent to the security analysts, so they know which features require more attention.

5 EVALUATION

In this section we present the experimental results evaluated on two network-based attack detection datasets. The first relies on characteristics found in individual network packets, while the second dataset extracts features across packets in a network flow and several temporal windows.

5.1 Datasets

The dataset for network packet-level features is extracted using the Gotham Testbed [18], a platform to create reproducible and large-scale IoT scenarios for dataset generation. The data processing and the FL anomaly detection model training methodology is taken from [40], where, for each network packet, 69 features are extracted from the packet header, content and inter-arrival times.

Gotham includes, among others, the real Mirai malware and several red-teaming tools to generate the attack dataset. For the evaluation in this paper, the selected behaviors and attacks generated with Mirai comprise most stages from the botnet life cycle (which includes stealthier as well as volumetric activities): C&C communication, network scanning for vulnerable devices, credential brute forcing, reporting victims to the C&C server, infecting the victims with the Mirai binary and remote command execution. DoS attacks are included in the following flow-level dataset. Red-teaming tools include activities such as Masscan and Nmap network-wide scans with different packet rates and port ranges, and CoAP amplification attacks.

Additionally, we use N-BaIoT [19] to evaluate the proposed method in a dataset based on network flow-level features. When a packet arrives/leaves, the feature extraction process computes a total of 115 features, which includes summary statistics taken over several temporal windows of packets and aggregated by different combinations of source IP, MAC and port addresses. Further details on the feature extraction process are given by Mirsky et al. [7]. For the attack evaluation, N-BaIoT includes two real IoT malware binaries, Gafgyt (a.k.a. BASHLITE) and Mirai, that generate the following volumetric attacks. For Gafgyt: Scan (network scanning for vulnerable devices), Junk (sending spam data), UDP flooding, TCP flooding and Combo (combination of Junk and opening connections to specific hosts). For Mirai: Scan, ACK flooding, SYN flooding, UDP flooding, UDPplain (UDP flooding with higher packet rate).

5.2 Federated learning model training

The federated model training corresponds to the first block depicted in Figure 1 (anomaly detection model training). The selected anomaly detection model for both datasets is an autoencoder trained and tuned on benign instances from their respective datasets and evaluated on data not used for training.

For the packet-level dataset, we use the same autoencoder described by Sáez-de-Cámara et al. [40], with input and output sizes of 69 nodes and 2 hidden encoder layers composed of 34 and 17 nodes, respectively. The decoder part is symmetric. The *ReLU* activation function is used after each layer. The model is trained in a FL setting composed of 11 clients, corresponding to one *City power* and ten *Combined cycle* nodes from [18], which use CoAP as the primary protocol to transmit the telemetry data. FL is performed for 100 rounds and 4 local training epochs using the Adam optimizer with a 0.005 learning rate and 1×10^{-5} L_2 regularization weight.

The anomaly threshold for each device is computed using another separate validation set of benign instances (also local to each device and not used during training), the maximum value of the autoencoder reconstruction error is selected as a threshold to minimize the number of false positives. After evaluating the FL model on the attacking instances, we obtain F1 scores greater than 0.9999.

For the flow-level dataset, we reproduce the autoencoder model from [19]. The autoencoder has an input and output size of 115 nodes and 4 hidden encoder layers composed of 86, 57, 37 and 28 nodes, respectively, with a symmetric decoder. We use the *ReLU* activation function after each layer. For the FL training, we select 2 clients, the two Doorbell IoT devices *Danmini* and *Ennio* from [19]. Features are transformed using a MinMax scaler fitted across the federated network. FL is performed for 30 rounds and 1 local training epoch using the Adam optimizer with a 0.008 learning rate and 1×10^{-5} L_2 regularization weight.

In this case, the anomaly detection threshold for each device is selected in the same way as described by Meidan et al. [19], taking the sum of the reconstruction error mean and standard deviation over a separate validation set of benign instances not used during training (scaled using the previously fitted MinMax scaler). The FL model evaluation on the scaled attack samples gives F1 scores greater than 0.9997.

5.3 Federated learning SHAP explainer and SHAP values

In this section, we are going to show the application of the FL SHAP explainer training and the generated explanations. These results correspond to the "Train XAI explainer" and "Generate explanations" steps from the third block shown in Figure 1.

As noted in Algorithm 1, the FL Kernel SHAP explainer model training requires a set of clients Z, the local number of clusters for each client $k^{(z)}$ and the number of global clusters k. Since the computation time increases linearly with the size of the background data k, we are going to select small values for k relative to the available number of training samples and set $k^{(z)} = k \; \forall z \in Z$ to simplify the parameter selection for Algorithm 1. However, since the selection of k can affect the generated SHAP values, we are going to repeat the process for two values, $k = 5$ and $k = 20$, to explore their effect.

After sampling the k background values and using them as a baseline to create the Kernel SHAP explainer (Algorithm 1), each client evaluates the explainer on the identified anomalous samples, and then, the SHAP values are normalized (lines 2–3 in Algorithm 2).

For the packet-level dataset, among the 11 clients used for the federated model training, 2 of them received attacks. Each attacked device is exposed to different anomalous activities; however, some are common to multiple devices. The first device is exposed to Mirai C&C traffic and the initial stages of the malware (scanning, preinfection and infection phases). The second device received various scanning activities from Nmap first, and then it was exploited to perform reflected DoS CoAP amplification attacks.

The generated SHAP values of the anomalies are local to each client; however, for illustrative purposes, Figure 2 shows, for both $k = 5$ and 20, a 2D visualization of the SHAP values of all the anomalous samples in a centralized way using the UMAP dimension reduction technique. In practice, centralizing the data would not be feasible in federated settings because it requires each client to transmit the SHAP values to the central server. Figure 2 highlights the difference between using $k = 5$ samples as baseline (Figure 2a) and $k = 20$ samples (Figure 2b). We use the same UMAP random seed initialization for both cases to make them comparable. The $k = 20$ case shows more clearly defined clusters compared to $k = 5$. Each anomalous point is colored according to an attack label (the *normal* label represents a few false positives). The labeling process is performed using a heuristic based on the IP source and origin addresses and timestamps, and it is only used for visualization purposes and not for training. Under the same label, there might be more than one anomalous behavior, and different labels can also have patterns in common, as shown in Figure 2.

We use the same methodology for the flow-level dataset as in the previous one. However, in the N-BaIoT dataset, for each device, the attack samples are provided in a separate file for each distinct attack type. The attacks for the *Danmini* device include 5 Gafgyt and 5 Mirai attacks, whereas, for *Ennio*, it only includes 5 Gafgyt attacks. Therefore, to train the Kernel SHAP explainer and the generation of the SHAP values, we are going to simulate 15 clients in the federated network, where each attack file is assigned to a simulated IoT client. To compute the SHAP background samples, the benign instances from *Danmini* are shared among the 10 simulated clients, and the benign instances from *Ennio* are shared for the remaining 5. Each simulated client then computes the SHAP values of its corresponding attack type anomalies (all simulated clients use the same trained FL anomaly detection model described in section 5.2).

Figure 3 shows the 2D visualization of the SHAP values (all centralized) for $k = 5$ (Figure 3a) and $k = 20$ (Figure 3b). In this case, the difference between the sizes of the SHAP background samples is not as apparent as in the packet-level dataset. The visualization shows interesting patterns in the SHAP values of the anomalous samples. For instance, Gafgyt Junk and Combo are close to each other and span a similar region in the embedding. According to [19], Gafgyt Combo comprises Gafgyt Junk and additional connections. Similarly, Gafgyt TCP and UDP share the same space in the embedding, both are attacks with similar behavior, but the feature extraction process does not distinguish between TCP and UDP. Mirai scan and Gafgyt scan activities are also placed in a similar embedding space.

5.4 Federated learning anomaly clustering

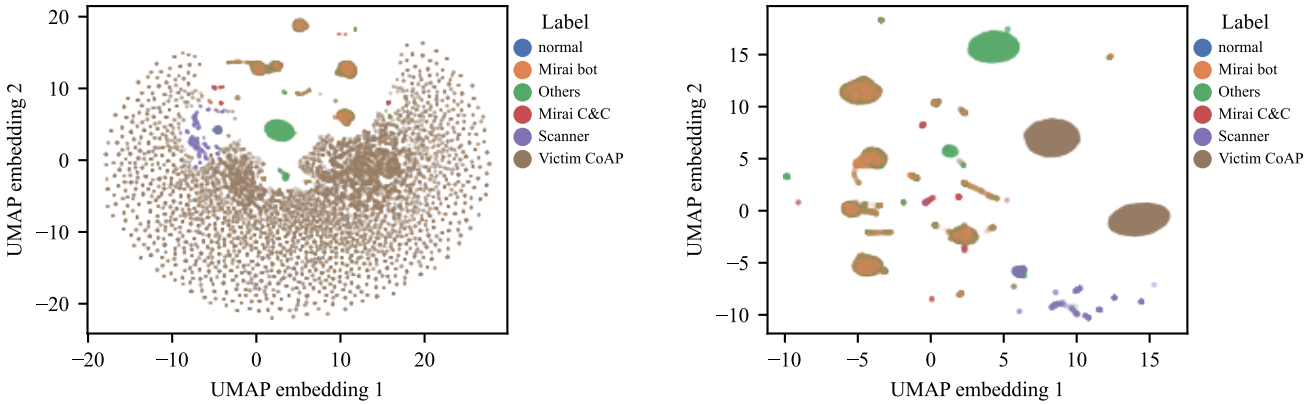
The federated anomaly clustering step is going to be performed using the SHAP values obtained with the $k = 20$ background samples case for both datasets. As noted in Algorithm 2, we estimate each $k^{(z)}$—the number of anomalous clusters local to each device—using HDBSCAN and compute k' at the server as the maximum of all the received $k^{(z)}$. We use the same HDBSCAN parameters for all the clients: minimum cluster size set to 300, min number of samples to 1 and cluster selection epsilon to 0.05.

For the packet-level dataset, the estimated number of clusters for the first client is $k^{(1)} = 14$, and $k^{(2)} = 4$ for the second. Thus, the final value for k' is set to 14 for both clients when performing the federated k-means process. Since k—the optimal value of the total number of anomalous clusters over the federated network—is unknown, we will perform multiple federated k-means trials for k ranging from k' to $(k' \times \text{number of devices}) - 1$ (from 14 to 27) and compute the corresponding CH scores, as explained in section 4.2. Additionally, we will perform 30 repetitions for each trial to account for the effects caused by the random initialization of the k-means centroids.

The obtained clustering validation metrics are shown in Figure 4. The results of the unsupervised internal validation metrics using the CH score computed in a federated (distributed) way are shown in Figure 4a, where higher scores indicate a better fit. Figure 4b shows an additional experiment to measure the clustering quality results between the federated and centralized settings. For this comparison, we compute the adjusted Rand score between the federated k-means and the centralized HDBSCAN on the joined data using the same parameters as for the estimation of the $k^{(z)}$ values for each client. The global HDBSCAN clustered the data into 16 clusters (and some non-clustered samples, which are considered noise). However, unlike HDBSCAN, k-means does not consider any samples as non-clustered noise.

In a deployment FL scenario, using ground truth clustering results or centralizing all the data is not feasible for selecting the optimal value for k. Accordingly, the decision will be only based on the unsupervised internal validation metrics, selecting the smallest number of clusters that show a high enough CH score. According to Figure 4a, $k = 22$ is an acceptable value. The final distribution of the number of anomalous samples for each cluster and client is detailed in Table 1. For each client, the table shows the percentage of the anomalous samples grouped into a particular cluster.

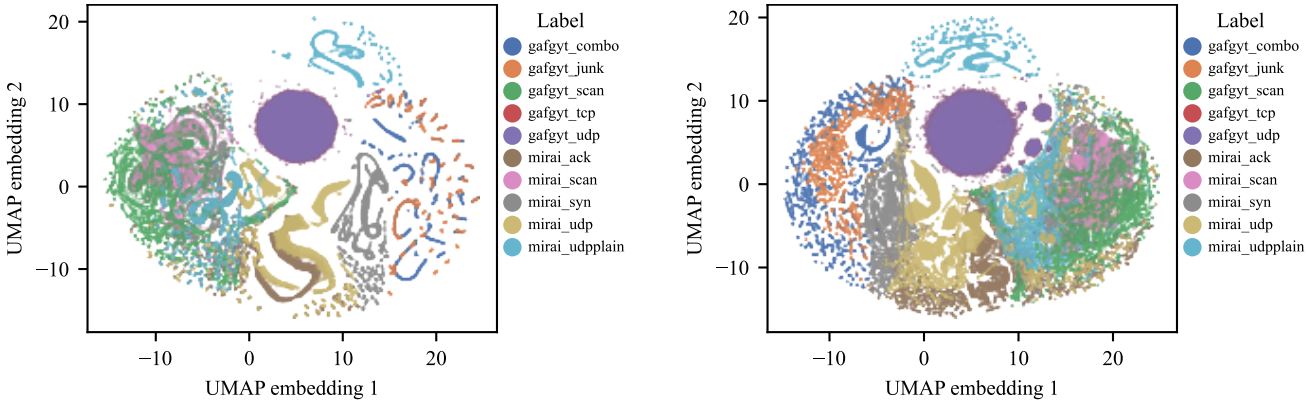
For the flow-level dataset, we follow the same methodology as in the previous case and use the same HDBSCAN parameters to estimate k', yielding a value of 8. This time we test k from 8 to 59 to



(a) SHAP using $k = 5$ samples as background.

(b) SHAP using $k = 20$ samples as background.

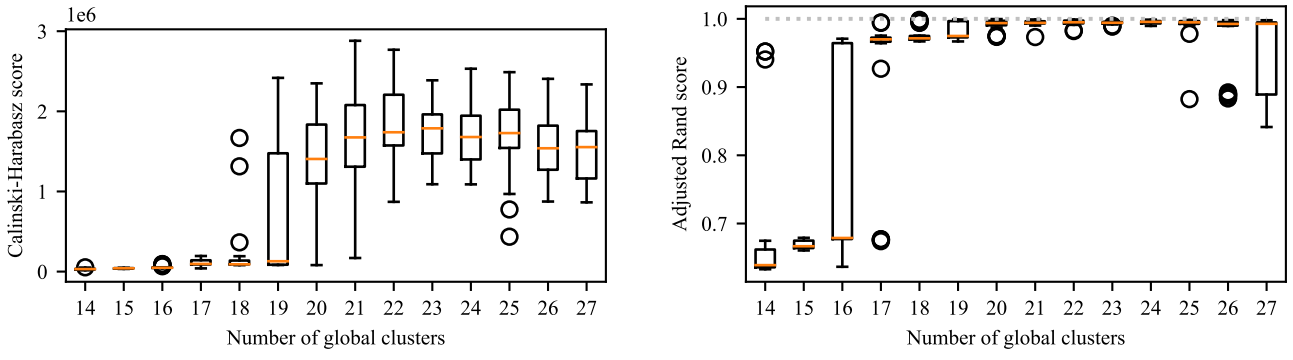
Figure 2: 2D visualization of the packet-based SHAP values of anomalous samples (centralized) total: 138,435 anomalies.



(a) SHAP using $k = 5$ samples as background.

(b) SHAP using $k = 20$ samples as background.

Figure 3: 2D visualization of the flow-based SHAP values of anomalous samples (centralized) total: 1,285,084 anomalies.



(a) Calinski-Harabasz score computed in a federated way.

(b) Adjusted Rand score between the federated k-means and centralized HDBSCAN clustering.

Figure 4: Federated k-means clustering validation metrics for the packet-based dataset. Horizontal axis represents the global number of clusters k. For each k, the box plot shows the scores for 30 repetitions.

Table 1: Distribution of the 22 global clusters for the packet-based dataset across the 2 clients that received attacks. The values are shown as percentage (%) of samples that belong to each cluster per client. A value of '-' represents 0 samples.

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21
Client 1	3.47	12.2	16.6	31.3	1.08	0.87	0.87	11.1	0.94	1.04	0.70	0.87	18.3	0.66	0.00	-	-	-	-	-	0.01	-
Client 2	-	-	-	-	-	-	-	-	2.70	-	0.00	-	-	-	47.1	0.15	47.1	0.02	2.44	0.01	0.06	0.33

reduce the number of repetitions. Figure 5 shows the results of the clustering validation metrics. Based on the results from Figure 5a, we select 11 as the global number of clusters. While the CH score seems to have an increasing trend for higher values of k, 11 is the smallest number of clusters that show a spike in the score. The final distribution of the clusters for each client is shown in Table 2.

5.5 Anomaly cluster alert explanation

Here we show the interpretation or explanation of the results obtained after the federated k-means clustering of the SHAP values from the anomalous samples from Table 1 and Table 2.

For the packet-level dataset (Table 1), we can see little overlap in the anomaly clustering results between the two clients, which is reasonable considering the different types of attacks that target the two clients. However, there is a significant overlap in the anomalies belonging to cluster C8. The most salient features given by SHAP that contribute towards classifying the packets as anomaly are `ip_tos`, `ip_flag_DF`, `sport_PRIVILEGED_PORTS`, `dport_PRIVILEGED_PORTS` and `ip_proto_ICMP`. The packets corresponding to C8 from both clients are composed of ICMP destination unreachable messages as a response to some port scanning activity.

The SHAP values corresponding to the nearest anomalous sample to each cluster center for the first client in the packet-based dataset is shown in the heat-map from Figure 6. The remaining heat-maps are all shown in the appendix in Figure 7 and Figure 8 for the first (again) and second clients, respectively.

C4 and C10 are related to the Mirai binary downloading stage from the first client. The second client also has a few packets in C10, which correspond to port scanning in the HTTP range. Most clusters C1-C3, C5-C7, C9 and C11-C13 are related to Mirai port scanning activities.

Some interesting clusters in client 2 are C14 and C16, which correspond to CoAP amplification attacks that send a flood of GET requests to the `.well-known/core` resource with a spoofed source address using code from the AMP-Research [41] tool. The legitimate training data of this particular device does include packets with the same request; however, the packets from the attack are correctly classified as anomalous. In particular, the anomalies from C16 show high SHAP values in the `ip_ttl` and `ip_flag_DF`. After inspecting the source code of the attack from [41], those fields are specifically set to certain values, which differed from the normal behavior, and the model detected those implementation particularities.

For the flow-level dataset (Table 2), we can see that clients do share samples from many clusters. In particular, the different activity from Gafgyt across the two IoT devices (*Danmini* and *Ennio*) show very similar distribution, while Mirai related attacks show different set of clusters, except for Mirai scan, which is similar to Gafgyt scan activity.

The SHAP values corresponding to the nearest anomalous sample to each cluster center are shown in the appendix from Figure 9 to Figure 23 for all 15 clients.

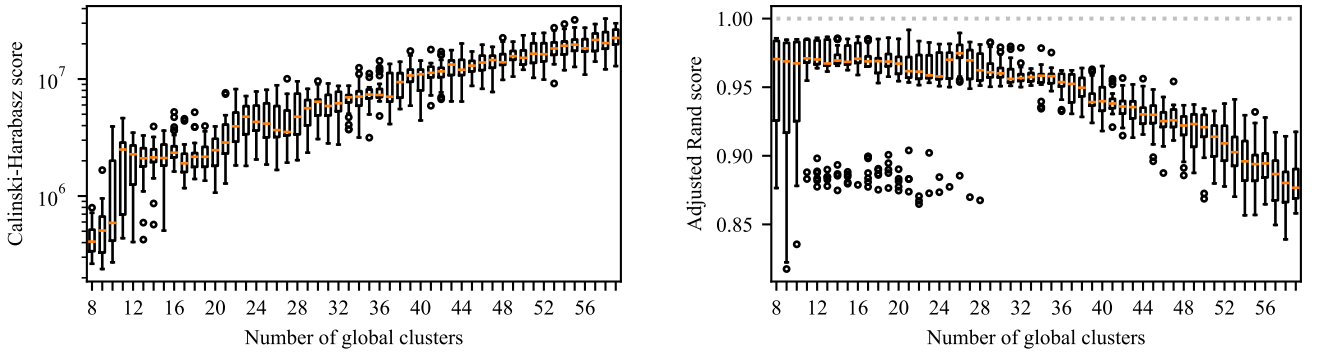
5.6 Anomaly message exchange

Listing 1 shows an example of an IDMEF alert message generated as a response to many anomalous samples from client 2 in the packet-based dataset falling under cluster C16. In addition, the message includes the `CorrelationAlert` class referencing another alert message of anomalies co-occurring in time that belong to another cluster center. The `AdditionalData` class is populated with extra information, such as the number of anomalies included in the alert and summary statistics (including mean, variance and percentiles) of the features of the data taken over all anomalous samples in the referenced cluster.

Listing 1: IDMEF alert message example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE IDMEF-Message PUBLIC
 "-//IETF//DTD RFC XXXX IDMEF v1.0//EN" "idmef-message.dtd">
<IDMEF-Message>
  <Alert messageid="0000641857181624681000002A6D0001">
    <Analyzer analyzerid="fl-client-01"/>
    <CreateTime ntpstamp="0xe7c2d598.0x0">2023-03-20-T12:52:40Z
    </CreateTime>
    <DetectTime>2023-03-20-T12:30:51Z</DetectTime>
    <Source>
      <Node>
        <Address category="ipv4-addr">
          <address>192.168.0.200</address>
        </Address> </Node> </Source>
    <Target>
      <Node>
        <Address category="ipv4-addr">
          <address>192.168.20.10</address>
        </Address> </Node>
      <Service>
        <portlist>5683</portlist>
      </Service> </Target>
    <Classification text="anomalies from cluster C16"/>
    <CorrelationAlert>
      <name>anomalies from multiple clusters in short time </name>
      <alertident>0000641855856299251000002A620001 </alertident>
    </CorrelationAlert>
    <AdditionalData meaning="packet_length-std" type="real">0.0
    </AdditionalData>
    <AdditionalData meaning="packet_length-mean" type="real">63
    </AdditionalData>
    <!-- (...) More data omitted (...) -->
    <AdditionalData meaning="anomalies count" type="integer">32171
    </AdditionalData>
  </Alert>
</IDMEF-Message>
```

While the correlation alert we describe is only temporal and computed from anomalies generated at each device in isolation, more sophisticated correlation processes could be made at the SIEM level. Including alert correlation across clients, analyzing alert clusters that usually appear together that could be attributed to attacks from certain tools or malware by correlating with alerts triggered from other security solutions and indicators of compromise.



(a) Calinski-Harabasz score computed in a federated way.

(b) Adjusted Rand score between the federated *k*-means and centralized HDBSCAN clustering.

Figure 5: Federated *k*-means clustering validation metrics for the flow-based dataset. Horizontal axis represents the global number of clusters k. For each k, the box plot shows the scores for 30 repetitions.

Table 2: Distribution of the 11 global clusters for the flow-based dataset across the 15 clients that received attacks. The values are shown as percentage (%) of samples that belong to each cluster per client. A value of '-' represents 0 samples.

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Danmini_Doorbell gafgyt_combo	-	83.10	0.01	14.28	2.30	-	0.23	0.07	0.01	-	-
Danmini_Doorbell gafgyt_junk	-	60.49	0.02	32.56	6.59	-	0.18	0.16	0.01	-	-
Danmini_Doorbell gafgyt_scan	-	-	0.01	-	-	-	95.13	4.85	0.01	-	-
Danmini_Doorbell gafgyt_tcp	-	-	0.00	-	-	-	-	-	99.91	0.09	-
Danmini_Doorbell gafgyt_udp	-	0.00	0.02	-	-	-	-	-	99.91	0.03	0.03
Danmini_Doorbell mirai_ack	61.20	-	0.01	-	-	4.17	33.68	0.94	-	-	-
Danmini_Doorbell mirai_scan	-	-	-	-	-	-	99.99	-	0.01	-	-
Danmini_Doorbell mirai_syn	62.54	-	0.00	-	0.00	6.88	28.35	2.22	0.00	-	-
Danmini_Doorbell mirai_udp	62.84	-	-	-	-	2.59	33.79	0.77	0.00	-	-
Danmini_Doorbell mirai_udpplain	0.01	-	0.01	-	-	-	42.63	1.42	-	-	55.93
Ennio_Doorbell gafgyt_combo	-	88.05	0.01	11.62	0.00	-	0.24	0.07	0.01	-	-
Ennio_Doorbell gafgyt_junk	-	65.44	0.02	31.76	2.43	-	0.18	0.15	0.01	-	-
Ennio_Doorbell gafgyt_scan	-	-	0.01	-	-	-	95.62	4.32	0.06	-	-
Ennio_Doorbell gafgyt_tcp	-	-	0.00	-	-	-	-	-	99.92	0.08	-
Ennio_Doorbell gafgyt_udp	-	-	0.01	-	-	-	-	-	99.95	0.04	0.01

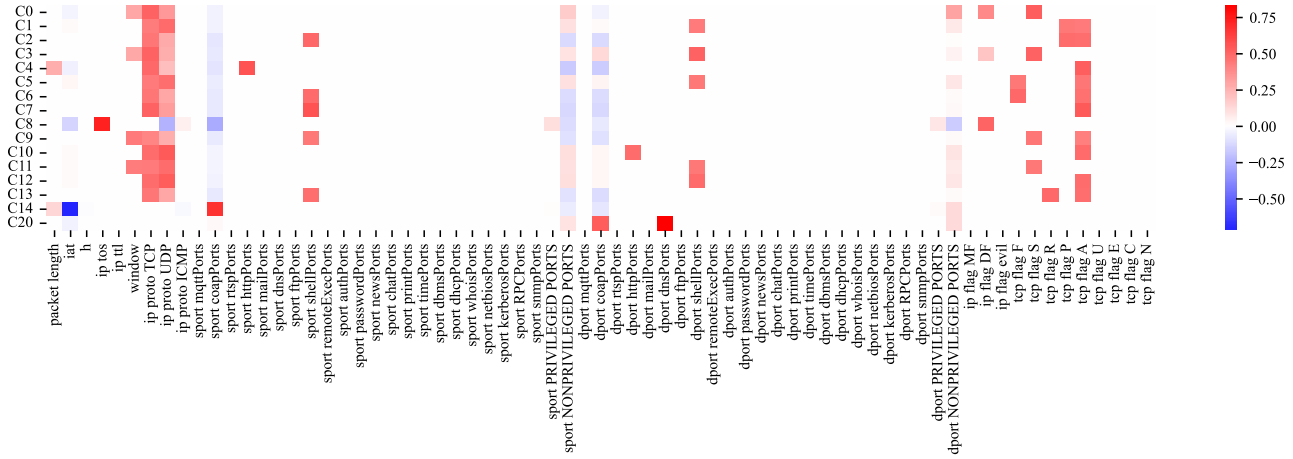


Figure 6: Client 1 SHAP values for each cluster center in the packet-based dataset.

The IDMEF data model includes other classes that could also be leveraged by the proposed system. One of those classes is the `Confidence` inside the `Assessment` class. The confidence could be assessed based on the distance of the detected anomalous samples to the centroid of it's corresponding cluster, density-based measurements or other types of fitness scores. Samples with low confidence scores in an alert could indicate that the network is facing new anomalous behaviors not observed during the training stages.

5.7 Algorithm costs

In this section, we provide the costs of the proposed approach in terms of data transmission in the FL setting, and also discuss the high computational requirements for SHAP.

Regarding the data transmission, here we provide expressions for the number of floats transmitted (uploaded and downloaded) across all the devices in the federated network. The actual bandwidth will depend on the float precision, compression and the data transmission protocol, which are not considered here as they are implementation dependent. Denoting as Z the number of participating clients in the federated network and d the dimensions of the data, the transmission costs for Algorithm 1 (SHAP baseline) are $2Z(kd + k)$, where here k refers to the number of SHAP baseline samples. For Algorithm 2, assuming that the number of anomalous clusters k is known, the cost is given by $2Z + Zk'd + Zkd$. Measuring the CH score (Algorithm 3) for a single value of k yields $Z(1 + d) + 2kZ$. Finally, by combining algorithms 2 and 3 with several trials to test different values for k (from some set of values $\mathcal{K}$, being T the size of that set) and R repetitions for each trial (as performed in the evaluation), the expression for the cost is $2Z + Zk'dTR + RZd \sum_{k_t \in \mathcal{K}} k_t + R \sum_{k_t \in \mathcal{K}} (Z(1 + d) + 2k_t Z)$.

Regarding the SHAP model, this paper focused on Kernel SHAP, as it is a general and model-agnostic approach to provide explanations. However, Kernel SHAP is computationally expensive, and the computation time increases linearly with the size of the background data. Nevertheless, other faster approaches exist for particular types of ML models, including Tree SHAP for trees and ensembles of trees, Deep SHAP or Gradient Explainer for many DL algorithms, or Linear Explainer for linear models [38]. Exploring and applying the proposed method for those model-specific approaches could be a relevant future line of work.

6 CONCLUSIONS

In this paper, we have proposed a methodology to explain and characterize anomalies of unsupervised intrusion detection models in a federated learning setting, where the clients throughout the network can have differences in data or behavior distribution and might also be exposed to distinct types of attacks. The explanations are based on the Kernel SHAP model-agnostic method, using a federated version of the k-means algorithm to subsample the background dataset required for SHAP model training across all the clients. We leverage the generated explanations by clustering (in the SHAP space) all the identified anomalies in the network using again an adapted version of the federated k-means algorithm. Since the number of anomalous patterns or groups is not known a priori, we also presented an adaptation of the Calinski-Harabasz internal cluster validation metric for distributed settings to allow the estimation of a suitable number of anomalous clusters found among all the clients.

A practical benefit of the proposed method is that all the federated steps can be performed in a one-shot manner (a single round of communication), which reduces the data transmission between the clients and the FL aggregation server. However, we note that selecting an adequate number of anomalous clusters requires repeating the federated k-means process for different values of k. Additionally, for robustness, it is recommended to perform various trials for the same k to account for random processes, such as the initialization of the centroids in the k-FED k-means process, as the experimental results show high variability in the Calinski-Harabasz scores. While each process requires minimal data transmission overhead proportional to k, multiple trials and repetitions can rapidly increase the cost; for communication efficiency, this should be considered compared to the amount of local training data on each device.

Both k-means and the Calinski-Harabasz algorithms tend to prefer isotropic cluster shapes as their main objective function is based on the minimization of the within-group sum of squares. However, some anomalous patterns might naturally cluster into elongated shapes. Studying and adapting other types of clustering and validation algorithms (such as density-based ones) to federated settings is a relevant line of future work.

In addition, in this work, we considered two different values for the number of SHAP background samples. Another future step could consider the identification of an optimal or suitable number of background samples required for each dataset to improve automation. Moreover, related to improved automation, developing methods or heuristics to reduce the search space for the HDBSCAN hyperparameters to estimate the local number of clusters, or the number of trials to identify a suitable number of anomalous clusters could help the automation and reduce communication costs.

Regarding privacy implications, a sensitive step is obtaining the SHAP background samples in a federated way, which includes disclosing few (much less than local training data amount) values of rounded k-means centroids to the aggregation server and back to the clients (although clients do not know which centroids came from which clients). Evaluating how much information is disclosed and, more interestingly, measuring how countermeasures such as differential privacy could be applied to the background samples and how it affects the generated explanations is a relevant future line of work to improve the robustness of the proposed methodology.

The proposed method identified several anomalous behaviors in the evaluated datasets and assigned a label to each of them that can be used to identify and characterize groups of anomalies. The labels are shared and known to all the clients and serve as a naming system to refer to the same anomalous patterns across all the clients in the federated network. New incoming alerts can be grouped and auto-labeled into the known anomaly behaviors, which can be used to send contextualized alerts representing multiple anomalies using the IDMEF message format, as shown in the results, for interoperability with third-party tools.

ACKNOWLEDGMENTS

The European commission financially supported this work through Horizon Europe program under the IDUNN project (101021911). It was also partially supported by the Department of Economic Development, Sustainability and Environment of the Basque Government under the ELKARTEK 2023 program, project BEACON (KK-2023/00085). Urko Zurutuza is part of the Intelligent Systems for Industrial Systems research group of Mondragon Unibertsitatea (IT1676-22), supported by the Department of Education, Universities and Research of the Basque Government.

REFERENCES

[1] F. Skopik, M. Landauer, and M. Wurzenberger, "Blind spots of security monitoring in enterprise infrastructures: A survey," *IEEE Security & Privacy*, vol. 20, no. 06, pp. 18–26, nov 2022.

[2] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[4] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A Survey of Deep Learning Methods for Cyber Security," *Information*, vol. 10, no. 4, Apr. 2019.

[5] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Feb. 2020.

[6] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the cicids2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*, 2021, pp. 7–12.

[7] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-3_Mirsky_paper.pdf

[8] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "Insomnia: Towards concept-drift robustness in network intrusion detection," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 111–122. [Online]. Available: https://doi.org/10.1145/3474369.3486864

[9] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492

[10] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*. IEEE Computer Society, 2010, pp. 305–316. [Online]. Available: https://doi.org/10.1109/SP.2010.25

[11] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[12] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale, "Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 112 392–112 415, 2022.

[13] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2022.

[14] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–17, 2022.

[15] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, "A survey on iot intrusion detection: Federated learning, game theory, social psychology and explainable ai as future directions," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[16] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[17] D. K. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 2611–2620. [Online]. Available: https://proceedings.mlr.press/v139/dennis21a.html

[18] X. Sáez-de-Cámara, J. L. Flores, C. Arellano, A. Urbieta, and U. Zurutuza, "Gotham testbed: A reproducible iot testbed for security experiments and dataset generation," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–18, 2023.

[19] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[20] X. Sáez-de-Cámara. (2023) Source code repository for: Federated explainability for network anomaly characterization. Accessed 2023/06/27. [Online]. Available: https://github.com/xsaga/federated-xai-anomalies

[21] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 364–379. [Online]. Available: https://doi.org/10.1145/3243734.3243792

[22] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73 127–73 141, 2020.

[23] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach, "Explaining anomalies detected by autoencoders using shapley additive explanations," *Expert Systems with Applications*, vol. 186, p. 115736, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421011155

[24] H. Liu, C. Zhong, A. Alnusair, and S. R. Islam, "Faixid: A framework for enhancing ai explainability of intrusion detection results using data cleaning techniques," *Journal of Network and Systems Management*, vol. 29, no. 4, p. 40, 2021. [Online]. Available: https://doi.org/10.1007/s10922-021-09606-8

[25] D. Rao and S. Mane, "Zero-shot learning approach to adaptive cybersecurity using explainable AI," *CoRR*, vol. abs/2106.14647, 2021. [Online]. Available: https://arxiv.org/abs/2106.14647

[26] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 91–99.

[27] K. S. K. Liyanage, Z. Tian, D. M. Divakaran, M. C. Chan, and M. Gurusamy, "Apex: Characterizing attack behaviors from network anomalies," in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2022, pp. 207–216.

[28] P. Barnard, N. Marchetti, and L. A. DaSilva, "Robust network intrusion detection through explainable artificial intelligence (xai)," *IEEE Networking Letters*, vol. 4, no. 3, pp. 167–171, 2022.

[29] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy, "Adept: Detection and identification of correlated attack stages in iot networks," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6591–6607, 2021.

[30] R. Haffar, D. Sánchez, and J. Domingo-Ferrer, "Explaining predictions and attacks in federated learning via random forests," *Applied Intelligence*, vol. 53, no. 1, pp. 169–185, 2022. [Online]. Available: https://doi.org/10.1007/s10489-022-03435-1

[31] T. T. Huong, T. P. Bac, K. N. Ha, N. V. Hoang, N. X. Hoang, N. T. Hung, and K. P. Tran, "Federated learning-based explainable anomaly detection for industrial control systems," *IEEE Access*, vol. 10, pp. 53 854–53 872, 2022.

[32] H. Yuan, M. Liu, L. Kang, C. Miao, and Y. Wu, "An empirical study of the effect of background data size on the stability of shapley additive explanations (SHAP) for deep learning models," *CoRR*, vol. abs/2204.11351, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.11351

[33] E. Albini, J. Long, D. Dervovic, and D. Magazzeni, "Counterfactual shapley additive explanations," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1054–1070. [Online]. Available: https://doi.org/10.1145/3531146.3533168

[34] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019. [Online]. Available: http://arxiv.org/abs/1912.04977

[35] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[36] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[37] A. Nadeem, D. Vos, C. Cao, L. Pajola, S. Dieck, R. Baumgartner, and S. Verwer, "Sok: Explainable machine learning for computer security applications," *CoRR*, vol. abs/2208.10605, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.10605

[38] S. M. Lundberg *et al.* Shap. a game theoretic approach to explain the output of any machine learning model. Accessed 2023/02/14. [Online]. Available: https://github.com/slundberg/shap

[39] B. Feinstein, D. Curry, and H. Debar, "The Intrusion Detection Message Exchange Format (IDMEF)," RFC 4765, Mar. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4765

[40] X. Sáez-de-Cámara, J. L. Flores, C. Arellano, A. Urbieta, and U. Zurutuza, "Clustered federated learning architecture for network anomaly detection in

large scale heterogeneous IoT networks," *Computers & Security*, vol. 131, p. 103299, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823002092

[41] AMP-Research. Research on exotic UDP/TCP amplification vectors, payloads and mitigations. [Online]. Available: https://github.com/Phenomite/AMP-Research

A APPENDIX

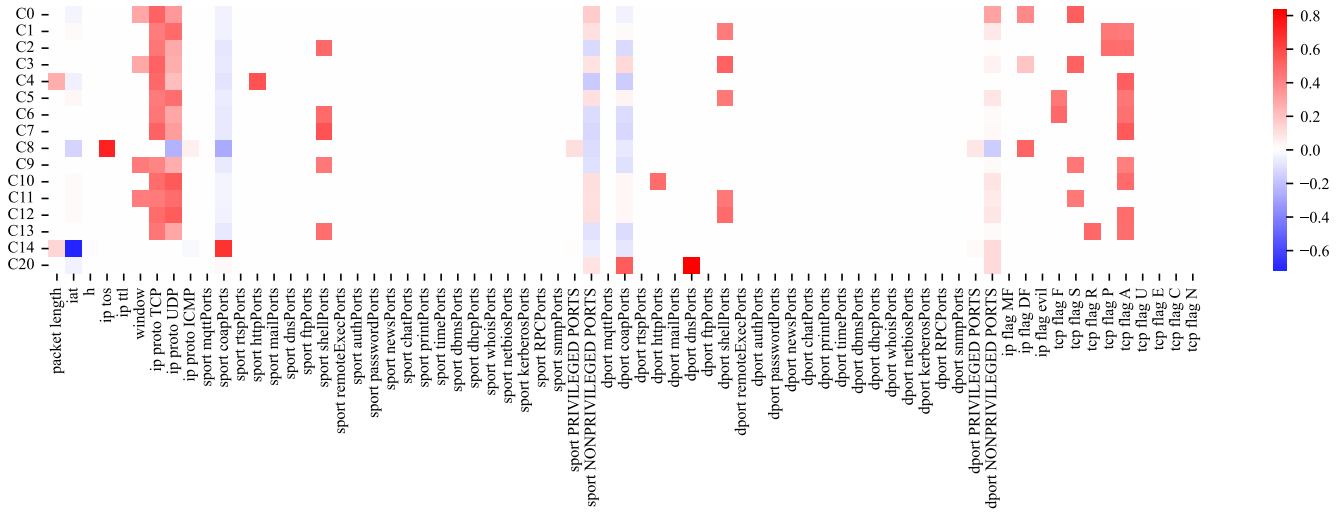


Figure 7: Client 1 SHAP values for each cluster center in the packet-based dataset. (Same as Figure 6)

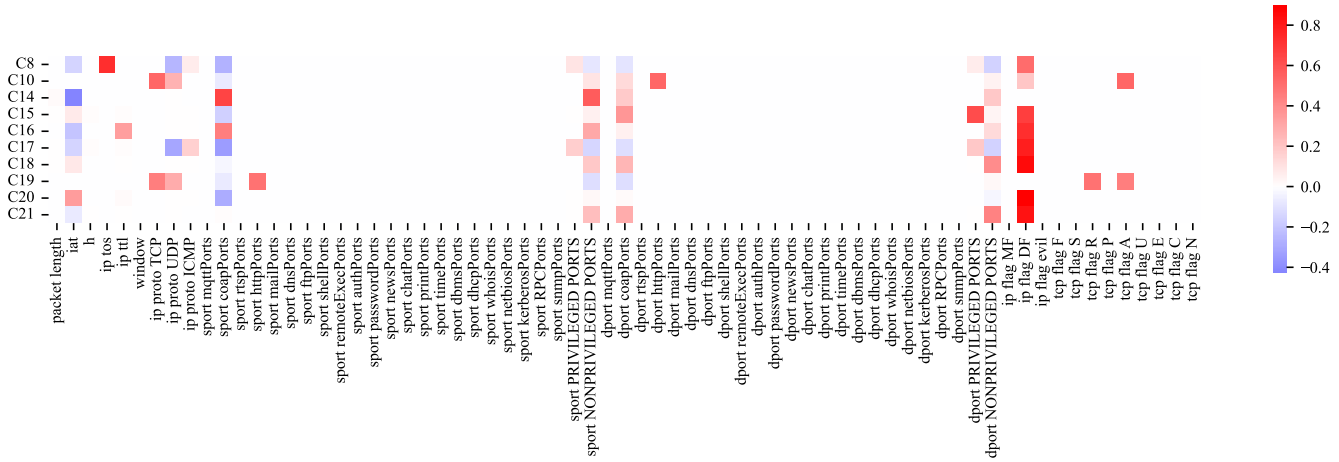


Figure 8: Client 2 SHAP values for each cluster center in the packet-based dataset.

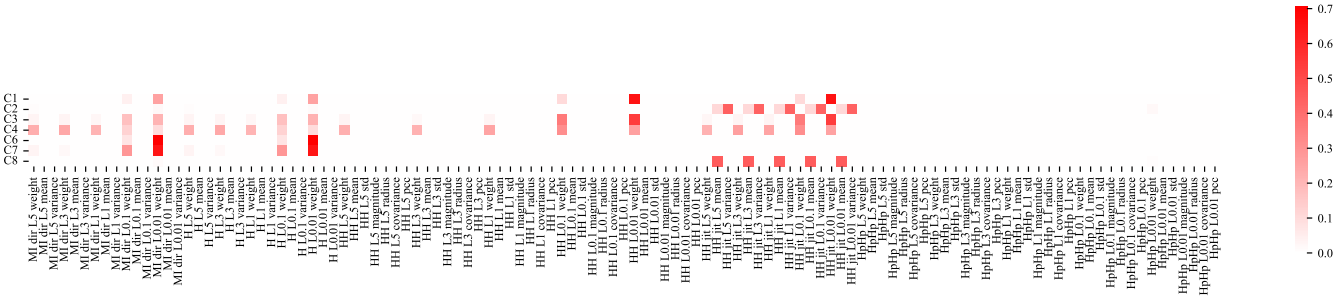


Figure 9: Client 1 SHAP values for each cluster center in the flow-based dataset.

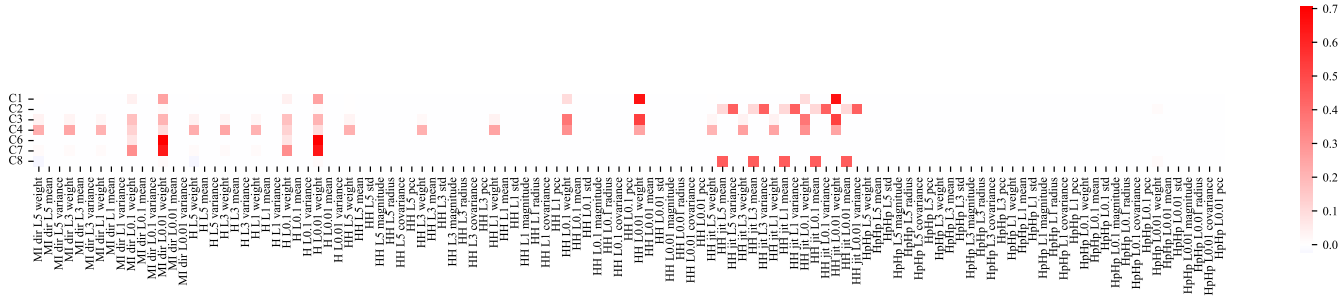


Figure 10: Client 2 SHAP values for each cluster center in the flow-based dataset.

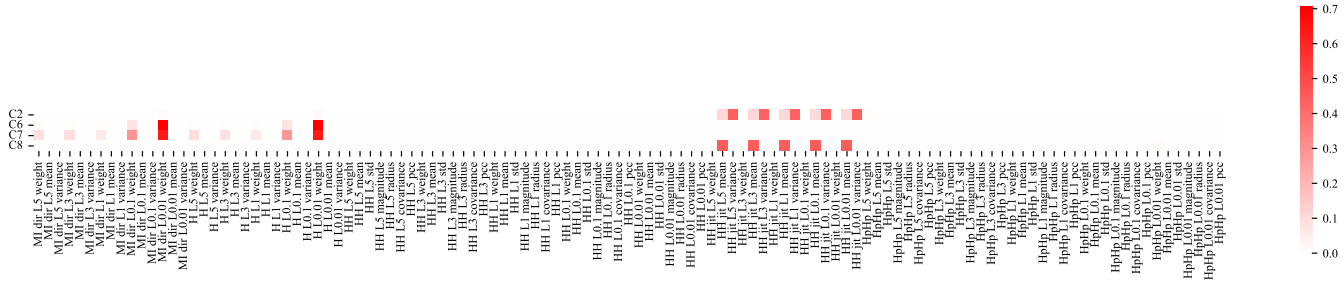


Figure 11: Client 3 SHAP values for each cluster center in the flow-based dataset.

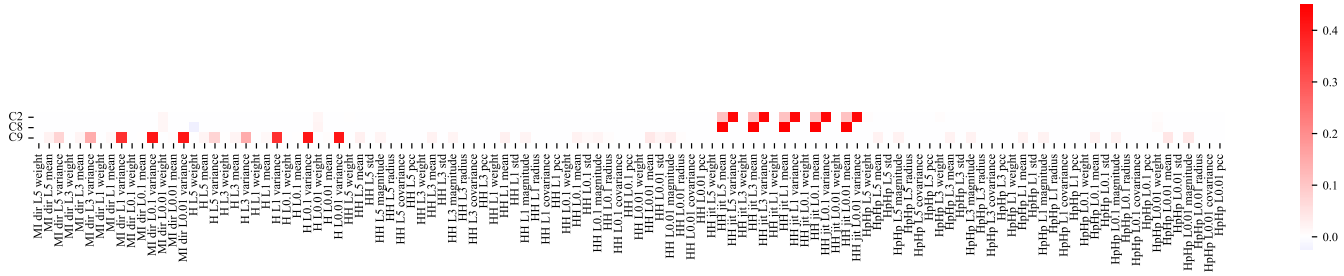


Figure 12: Client 4 SHAP values for each cluster center in the flow-based dataset.

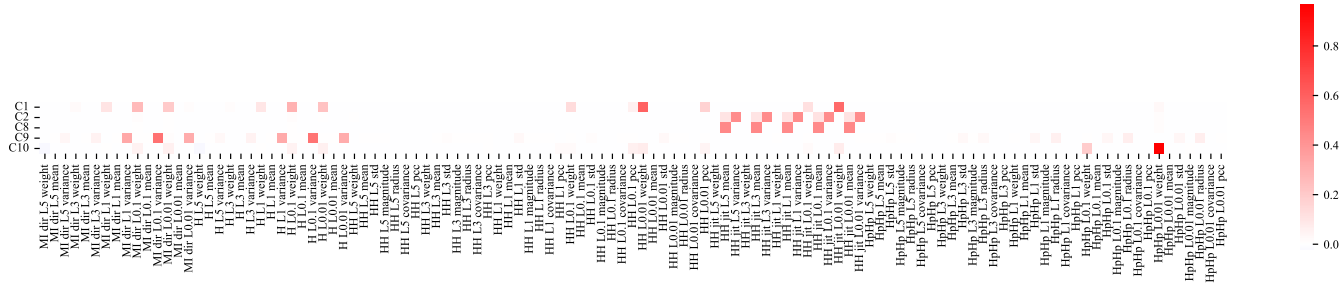


Figure 13: Client 5 SHAP values for each cluster center in the flow-based dataset.

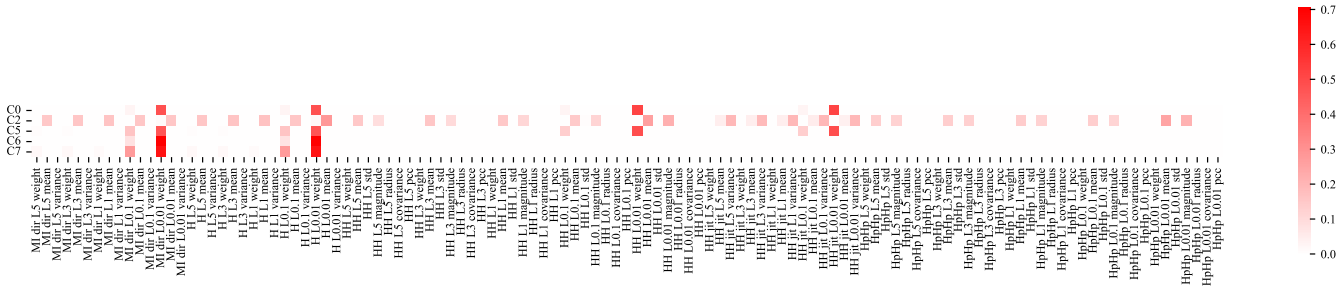


Figure 14: Client 6 SHAP values for each cluster center in the flow-based dataset.

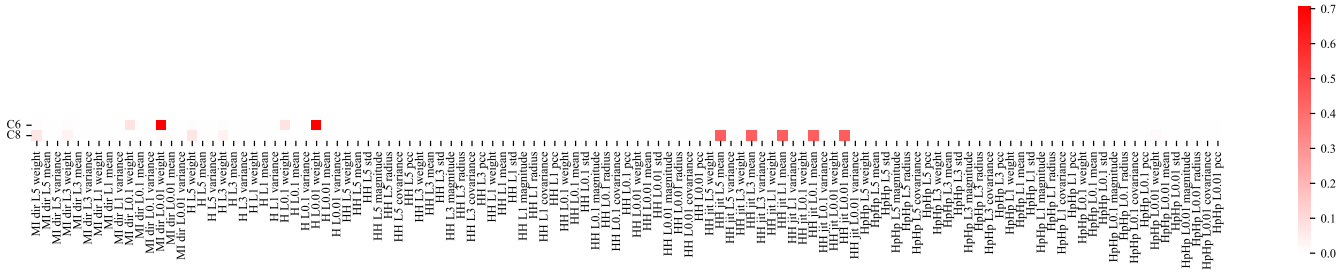


Figure 15: Client 7 SHAP values for each cluster center in the flow-based dataset.

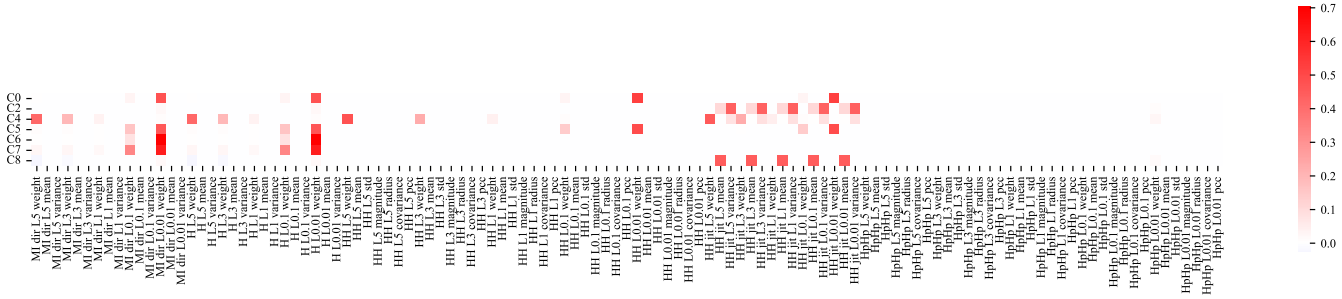


Figure 16: Client 8 SHAP values for each cluster center in the flow-based dataset.

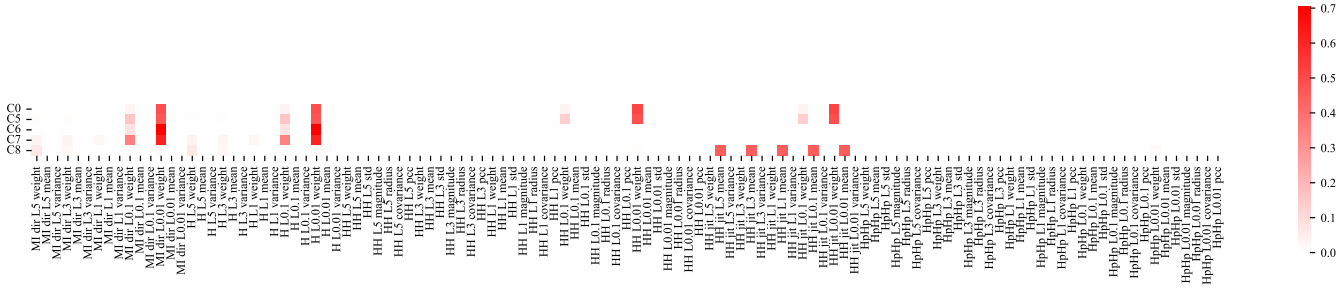


Figure 17: Client 9 SHAP values for each cluster center in the flow-based dataset.

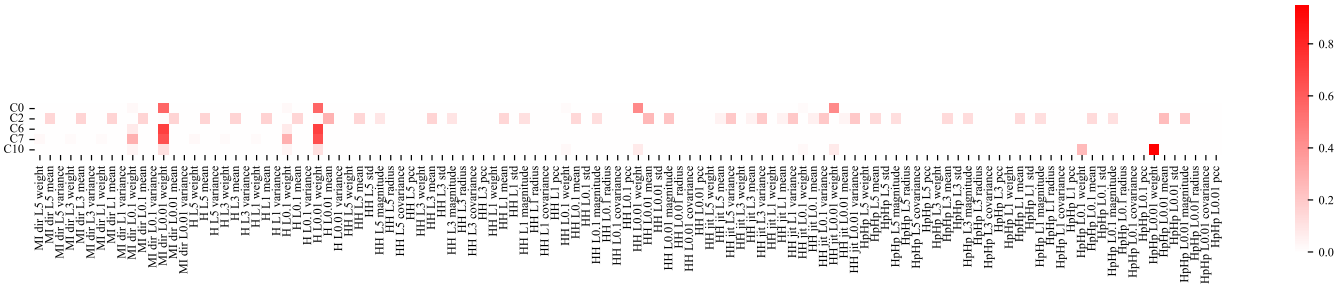


Figure 18: Client 10 SHAP values for each cluster center in the flow-based dataset.

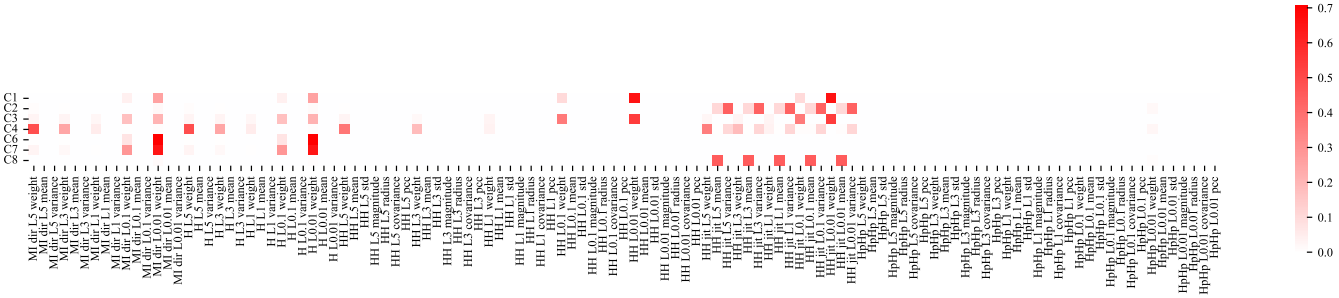


Figure 19: Client 11 SHAP values for each cluster center in the flow-based dataset.

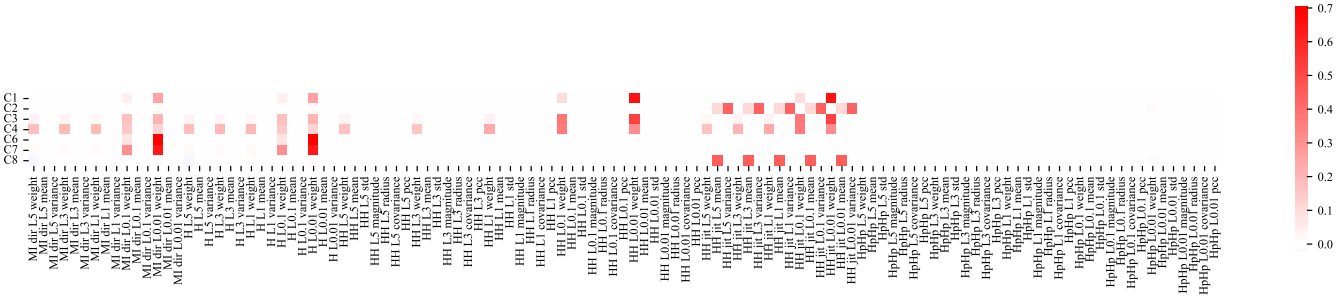


Figure 20: Client 12 SHAP values for each cluster center in the flow-based dataset.

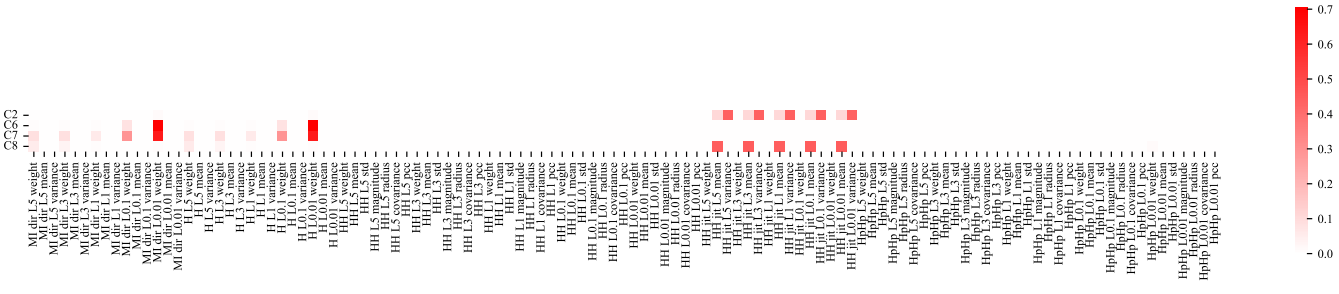


Figure 21: Client 13 SHAP values for each cluster center in the flow-based dataset.

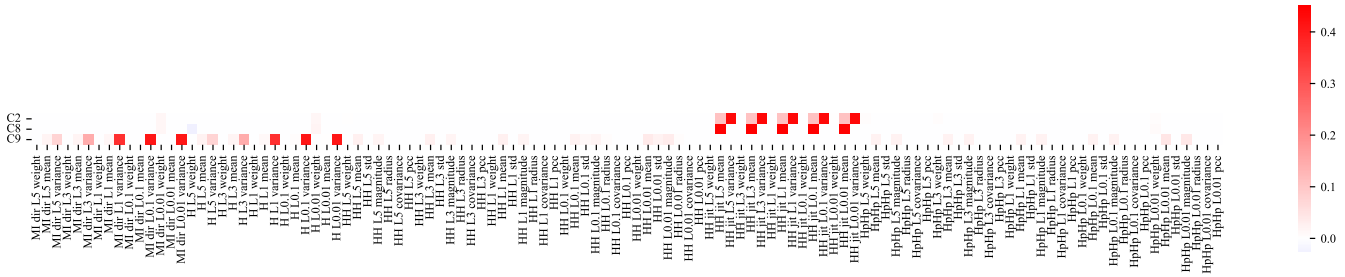


Figure 22: Client 14 SHAP values for each cluster center in the flow-based dataset.

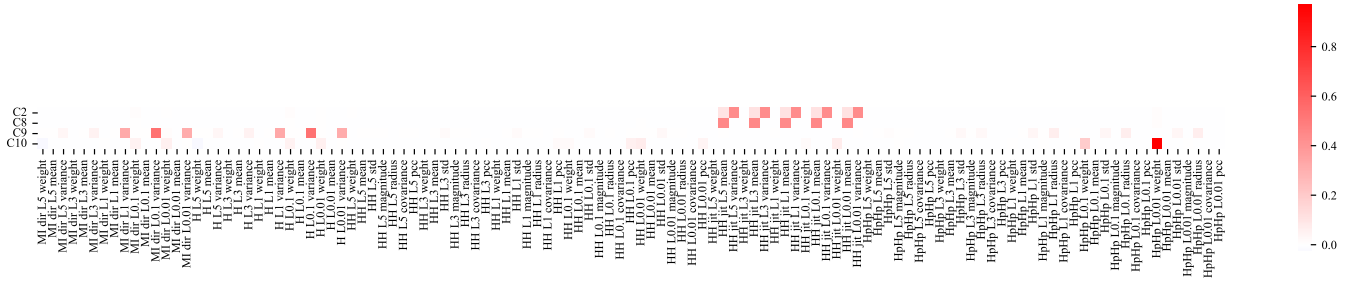


Figure 23: Client 15 SHAP values for each cluster center in the flow-based dataset.