

RESEARCH ARTICLE

Autonomous Federated Learning for Distributed Intrusion Detection Systems in Public Networks

ALIREZA BAKHSHI ZADI MAHMOODI¹, SAEID SHEIKHI, (Member, IEEE),
ELLA PELTONEN¹, (Senior Member, IEEE), AND PANOS KOSTAKOS¹

Faculty of Information Technology and Electrical Engineering, University of Oulu, 90014 Oulu, Finland

Corresponding author: Panos Kostakos (panos.kostakos@oulu.fi)

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Program (IDUNN Project) under Grant 101021911, and in part by the Academy of Finland 6Genesis Flagship Program under Grant 318927.

ABSTRACT The rapid integration of IoT, cloud, and edge computing has resulted in highly interconnected networks, emphasizing the need for advanced Intrusion Detection Systems (IDS) to maintain security. Successful AI-based IDS relies on high-quality data for model training. Even though a vast array of datasets from controlled settings are accessible, many fall short as they are outdated and lack the representative data of network traffic dynamics typically seen in public networks. This paper aims to advance understanding in designing testbed architectures for defense mechanisms within public networks. At its core, this research introduces a unique testbed utilizing the connectivity of panOULU Municipal public network in the city of Oulu, Finland. This experimental setup examines AI-driven security across the public network. It utilizes edge-to-cloud infrastructures, incorporating Software-Defined Networking (SDN) and Network Function Virtualization (NFV) via the VMware vSphere platform. During the training phase, a script distinguishes incoming packets as either benign or malicious based on well-defined local parameters and simulated attack scenarios. This labeled data is then utilized for training machine learning models within the Federated Learning framework, FED-ML. Subsequently, these models are evaluated on previously unseen data. The entire procedure, from traffic gathering to model training, operates without human involvement. The evaluation dataset and testbed configuration we have made publicly available through this research can deepen our understanding of the challenges in safeguarding public networks, especially those that blend various technologies in diverse environments.

INDEX TERMS Network security, cybersecurity, federated learning, data engineering, distributed computing, stream processing.

I. INTRODUCTION

As we journey into an era characterized by technological advancements, the surging growth of interconnected devices such as sensors, the Internet of Things (IoT), smartphones, and drones underscores the pressing need for a secure and trustworthy communication infrastructure. The constraints on capacity and latency inherent in the current 4G and 5G networks highlight the urgency for innovative solutions [47]. Consequently, cutting-edge technologies like network function virtualization (NFV), software-defined networking (SDN), edge/cloud computing, and collaborative federated

learning have become instrumental in shaping novel system architectures [14], [48], [53], [59]. This revolution signifies a shift from the conventional “network of networks” model to a forward-thinking “service of services” paradigm [53].

The ongoing evolution towards service-centric 6G networks necessitates the development of adaptable devices and systems attuned to this new connectivity landscape. At the same time, emerging technologies that enable a seamless fusion of virtual experiences with our physical reality, coupled with the Internet of Things increasingly interweaving itself into our homes, cities, and workplaces, are fundamentally reshaping how we engage with technology. The ongoing Industry 4.0 revolution further propels this integration of information technology (IT) and operational

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru¹.

technology (OT), giving rise to AI-enabled systems enhancing productivity and flexibility. Through the use of sensors, actuators, and software [10], IoT systems can be managed and controlled remotely, leading to faster manufacturing, enhanced customization, and the emergence of innovative business models [45].

Nonetheless, this progress also raises significant concerns. Critical National Infrastructures (CNIs), such as hospitals, ports, energy suppliers, and water distributors, which heavily depend on Supervisory Control and Data Acquisition (SCADA) or Industrial Control Systems (ICS) for management [20], are now increasingly susceptible to cyber attacks. Recognizing this vulnerability, political actors, including the European Union, have initiated security measures and policies to safeguard these systems [35]. Yet, to fully secure CNIs, it is crucial to introduce comprehensive security measures encompassing all legal, capacity-building, organizational, and technical aspects of cybersecurity. These include policy formulation, directives, regulations, and technical solutions to detect and prevent cyber threats.

The rapid rise of Networks and Information and Communications Technology (ICT) systems has exposed sensitive data to potential attacks from internal and external intruders [42]. High-profile security breaches, such as the breach experienced by Yahoo, leading to millions of dollars in losses [55], underscore the gravity of the problem. Advances in software, hardware, and network topologies, including those related to the Internet of Things, have enabled the development of increasingly sophisticated attack algorithms [57].

To mitigate these issues, a robust Intrusion Detection System (IDS) is essential to promptly identify and categorize attacks, breaches, and breaches of security protocols. IDS is a security technology that monitors network traffic and systems for malicious activities or policy violations. There are two types of IDS based on intrusive behaviours: the host-based intrusion detection system (HIDS) and the network-based intrusion detection system (NIDS) [38]. AI-powered cyber solutions specializing in threat hunting, intrusion detection, privacy protection, malware detection, and digital forensics can be a response to some of these challenges [10]. These centralised solutions rely on training datasets to assess their performance and make informed decisions.

There are three standard methods to analyze network traffic flows: stateful protocol analysis, anomaly detection, and misuse detection [58]. Each of these methods has its advantages and disadvantages. Anomaly detection, which detects unknown malicious activities through heuristics, has a high rate of false positives, making it less reliable [38]. On the other hand, misuse detection uses filters and signatures to detect known patterns of malicious traffic but requires regular updates to its signature database to detect newer threats. Alternatively, stateful protocol analysis is the most reliable method, as it uses predefined vendor settings to identify deviations from proper protocols and applications based on known protocols and applications [58].

Overall, acquiring valid training datasets to evaluate various proposed techniques presents a significant challenge due to the complexity of networks and systems, as well as the scarcity of high-quality testbeds that encompass both normal and malicious network traffic.

The research challenge we address in this paper involves creating a comprehensive, economic, adaptive security testbed characterization. The proposed testbed effectively addresses the challenges posed by the evolving landscape of mobile networks, interconnected systems, and emerging technologies while ensuring the privacy and resilience of users, devices, and critical infrastructures. Further, we define the four challenges (or research questions) addressed in this paper:

Challenge 1: The testbed characterization should be defined by enabling technologies, especially Network Function Virtualization (NFV), Software-Defined Networking (SDN), edge/cloud computing, and federated learning (FL).

Challenge 2: Federated learning (FL) should be effectively incorporated into intrusion detection systems (IDS) to enhance their accuracy, adaptability, and scalability while preserving the privacy and security of individual devices and data sources in distributed and heterogeneous network environments.

Challenge 3: It is crucial to establish realistic testing environments to assess robust security methods. This is especially true considering the intricate nature of public networks and the scarcity of high-quality, representative datasets.

Challenge 4: Need to define design principles for secure and distributed architectures for complex systems like IoT networks and address the protection of critical national infrastructures from cyber-attacks.

Contribution: Building on our prior research [36], this article introduces an orchestrated testbed infrastructure designed for cybersecurity experimentation to tackle the aforementioned research challenges. In the communication framework, the edge, fog, and cloud tiers seamlessly integrate, leveraging the panOULU Municipal public network that spans across the city of Oulu, Finland. The driving technology behind the fog layer is VMware vSphere, which is managed and orchestrated using vCenter Server Appliance. Functionalities such as Network Function Virtualization (NFV), Software-Defined Network (SDN) and Service Orchestration (SO) are enabled by utilizing on-premise hardware resources at the University of Oulu. These components facilitate the generation of datasets in both streaming and batch modes. The streaming data is used for learning, while the batch dataset is employed for evaluation.

Furthermore, we present a novel Distributed Network-based Intrusion Detection System using a FED-ML architecture that utilizes streaming high-quality data from diverse data sources across multiple tiers in the proposed architecture, interconnected by the panOULU public network. The streaming data, encompassing both regular and malicious

network traffic, is employed to train various ML models in a federated manner. In this architecture, nodes continuously collect streaming data containing network information, such as network packets. A custom-built agent is then responsible for processing the collected packets and labelling the data as regular or malicious based on the attackers' IP addresses within the network. The processed data is fed to FED-ML models in each learning round, with the learning process continuing until the desired number of rounds is satisfied. This entire process is autonomous and requires no human involvement. After the models are trained, they are tested on new, previously unseen data to assess their effectiveness.

II. PRIOR RESEARCH

Self-learning systems, including unsupervised, semi-supervised, and supervised machine learning algorithms, have been proposed as a potential solution to counter security threats. Despite the multitude of machine learning solutions available in scholarly literature, the assimilation of these strategies into commercially operated network-based Intrusion Detection Systems (NIDS) is still in its infancy [16], [17], [46], [58]. The main challenge with current machine learning solutions lies in their high false positive rates, which are often associated with low-quality publicly accessible training data, in addition to their substantial computational expenses [13].

Benchmark datasets are crucial in training ML/DL models for anomaly and intrusion detection. They are extensively used because of their significance in facilitating objective evaluation, replication, and validation of security solutions. The available public datasets, encompassing a broad spectrum of devices and setups, are classified into seven categories: Internet-connected devices, electrical networks, virtual private networks, Android applications, Internet traffic, IoT traffic, and network traffic. The most prominent of these datasets include DARPA [34], KDDCup99 [54], NSL-KDD [54], Kyoto [52], UNSW-NB15 [41], CIC-IDS2017 [50], CSE-CIC-IDS2018 [50], WSN-DS [11], DEFCON [44], CAIDA [3], CDX [2], CIC DoS [4], ISCX [5], ADFA2013 [1], TON_IoT [40], MAWI [6].

The KDDCup 99 dataset is commonly used in academic research related to intrusion detection. This dataset encompasses four categories of attacks, namely, R2L (Remote to Local), U2R (User to Root), DoS (Denial of Service), and Probe [58]. Based on the investigation by [61], the decision tree and Naive Bayes algorithms demonstrated strong performance on this dataset. Additionally, the efficacy of Naive Bayes for the given dataset is highlighted by [56]. Conversely, [7] indicated that using P-rules and N-rules demonstrated notable performance across a majority of categories on this dataset, with the exception of the U2R category.

Deep learning has been leveraged in various aspects of cybersecurity such as traffic analysis, network traffic forecasting, intrusion detection, and ransomware identification [57], [58]. Commonly utilized deep learning models in

intrusion detection systems for cybersecurity include: Convolutional Neural Network (CNN), Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN), Replicator Neural Network (RNN), Deep Neural Network (DNN), Feed Forward Deep Neural Network (FFDNN), Recurrent Neural Network (RNN), Deep Auto-Encoder (DAE), Deep Migration Learning (DML), and Self-Taught Learning (STL).

Sydney et al. [27] achieved an accuracy of 99.69% with 30 neurons in Feed-Forward Deep Neural Networks (FFDNN), utilizing the NSL-KDD dataset and implementing two-way normalization with feature transformation. Using Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN), Feng et al. [19] achieved an accuracy of 0.57 and 0.78 for detecting XSS attacks, respectively. By implementing Long Short-Term Memory (LSTM) within a Recurrent Neural Network (RNN), Kim et al. [30] achieved an accuracy rate of 0.988 on the KDDCup99 dataset. Similarly, Jiang et al. [24] used the same methodology and reported an accuracy rate of 0.9894 on the NSL-KDD dataset. Similarly, Basumallik et al. [15] claim an accuracy of 0.9867 for CNN with 512 fully connected neurons.

Salama et al. [49] improved their results by combining Support Vector Machine (SVM) and Restricted Boltzmann Machine (RBM) rather than solely relying on SVM. However, Alrawashdeh et al. [12] reported that their results surpassed those of Salama et al. [49] when they combined RBM with a Deep Belief Network.

Zhang et al. [60] combined a deep belief network with an improved genetic algorithm to achieve 0.99 detection rate for NSL-KDD dataset. Shone et al. [51] used an auto-encoder with non-symmetrical multiple hidden layers to achieve an accuracy of 0.9785 and 0.8542 on KDDCup99 and NSL-KDD, respectively. Javed et al. [23] used self-taught learning on NSL-KDD to achieve 0.7576 accuracy. Using deep migration learning, Li et al. [32] reached a detection rate of 0.9105 for KDDCup99. Cordero et al. [18] used replicator neural networks where entropy extraction includes an aggregation of packets, segmentation of the flows into time windows, and selection of features of interest from the flows.

Recently, research efforts have shifted towards testbed characterizations that more accurately represent the emerging threat landscape introduced by IoT networks. A notable example is the release of the TON_IoT dataset [40]. Originating from a distributed testbed architecture, this dataset seeks to shed light on the genuine behaviors of real-world IoT networks and emerging cyber threats. Such insights are essential for assessing the reliability of newly developed systems. Specifically, the introduced fog and edge computing layers offer on-site services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [39]. These layers enable end-users to handle their data or IoT systems, bestowing them with the advantage of local intelligence and analytics. This removes the need to transfer large volumes of data to the cloud [40].

Furthermore, fog and edge computing in the TON_IoT testbed differ primarily in the location of their computing, intelligence, and analytics components [39]. The edge layer situates these elements in physically sensible devices, including IoT devices, laptops, personal computers, and physical servers. Meanwhile, the fog layer positions its computing, analytics, and intelligence within Local Area Networks (LANs), routing data through gateways to reach their target destinations [39].

Although prior research has addressed some of the challenges highlighted in the introduction, no dataset emerges as a perfect representation [21], [43], [50]. The present research advances the field by leveraging real-time streaming data from the panOulu public network, applying it within a federated machine learning framework for intrusion detection. Additionally, it demonstrates the practical viability of this approach in a real-world distributed and decentralized deployment environment. The evaluation of the proposed architecture also involves analyzing the trained models using different performance metrics. The deliberate choice of Federated Learning (FL) as a technology in this work stems from its capability to tackle diverse challenges. FL offers benefits such as data privacy and ubiquitous intelligence, while facilitating seamless, flexible, and scalable communication with high performance. Moreover, it enables ultra-low latency and energy-efficient networking, making it suitable for supporting time-critical applications with dynamic requirements [8].

III. ARCHITECTURE

This section introduces the proposed architecture designed to produce a dataset that closely reflects contemporary edge networks and IoT devices utilized by smart cities and organizations in the current landscape. Additionally, we provide an overview of the federated learning techniques developed and integrated into this architecture.

The proposed architecture is based on diverse systems and hardware devices, encompassing IoT devices, laptops, cell phones, and more. It integrates layers of fog and edge networks to emulate the authentic real-world configuration of contemporary networks, showcasing the depth and sophistication of these systems. The architectural design is depicted in Figure 1.

In the proposed architecture, the edge layer encompasses IoT devices, servers, routers, laptops, Raspberry Pi etc., while the fog layer is designed for intelligence, computation, and analytics leveraging hypervisor technology over LAN. Furthermore, the cloud layer is used for hosting the Flower server — an open-source framework for federated learning systems. The Flower framework enables network streaming data to be processed on distributed devices or servers, eliminating the need to store vast amounts of data, while also supporting decentralized model training.¹ Flower clients can connect to the Flower server in the cloud to retrieve updates

from other peer Flower clients that have been trained on various streaming data sources.

This architecture is designed to produce streaming data that authentically mirrors the contemporary landscape of smart cities and connected places. Federated learning is implemented on top of this architecture to enable collaborative machine learning between devices in the Municipal public network while maintaining data privacy. This approach allows for the development of intelligent and efficient systems that can benefit various industries, including healthcare, transportation, and energy.

The key functionalities of each layer in the proposed architecture are as follows:

- The fog layer is a virtualized environment that controls services and virtual machines using the VMware vSphere platform. This platform enables the seamless integration of frameworks for Network Function Virtualization (NFV), Software Defined Networking (SDN), and Service Orchestration (SO) into the proposed architecture.
- These frameworks are made possible through SDN management and hypervisor technology.
- The physical devices are included in the edge as the infrastructure for deploying virtualization. This layer includes multiple IoT devices such as smartphones, smart TVs, cameras, air purifiers, etc., and host systems such as servers, workstations, and laptops.
- The cloud is specifically used to host the Flower server, which is required for federated learning. Therefore, all the Flower clients process their streaming data locally, and various machine learning models are trained in a decentralized manner. The learning weights of these individual ML models are sent to the Flower server in the cloud to be aggregated and averaged. Subsequently, the final output from the Flower server in the cloud is relayed to the Flower clients in the fog layer to ensure they remain updated. This entire process operates continuously and autonomously without the need for human intervention.

The proposed architecture leverages various technologies to enable flexibility and dynamism between layers, including Software Defined Networks (SDN), Network Function Virtualization (NFV), and Service Orchestration (SO). The effective operation of the proposed architecture hinges on the integral roles played by each of these technologies. For a more comprehensive elucidation of their interplay, we examine the functionalities inherent in each layer more in-depth.

- 1) Service Orchestration (SO) refers to coordinating and managing multiple services to achieve a specific business process or goal to establish and deliver end-to-end services [39].
- 2) Software Defined Networks (SDN) enables the development of virtual networks that possess the functionalities of physical networks [31]. SDN is employed

¹<https://flower.dev/>

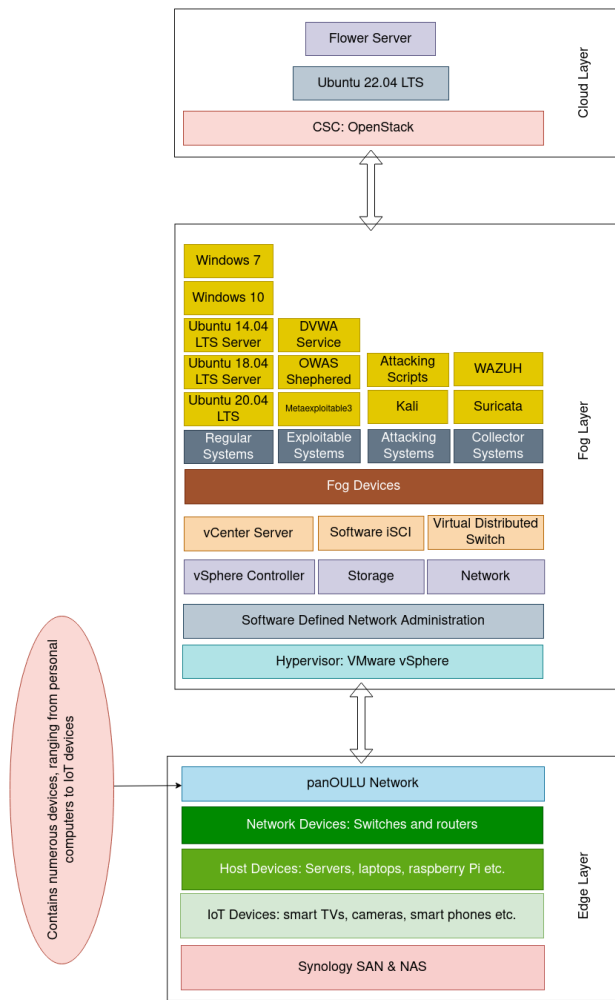


FIGURE 1. Proposed dynamic architecture for evaluating security applications across the edge-fog-cloud continuum.

to enhance network performance through dynamic and programmable network configurations.

- 3) Network Function Virtualization (NFV) facilitates virtualization and dynamic provisioning of network services by decoupling network functions from dedicated hardware [22].

Figure 2 illustrates the proposed architecture’s operational mechanism of federated learning. Each Suricata machine in the fog layer hosts an individual Federated Machine Learning (FED-ML) client. The FED-ML clients collect streaming data by storing network packets as pcap files. It is worth noting that no human intervention is required in the whole process. Following the preprocessing and labeling of the streaming network data, the FED-ML models engage in local training using this refined dataset. During each iteration of the procedure, every FED-ML client transmits the learned model’s weights to the Flower server located in the cloud. The blue and green dotted and dashed lines in Figure 2 illustrate the data transmission path within the architecture.

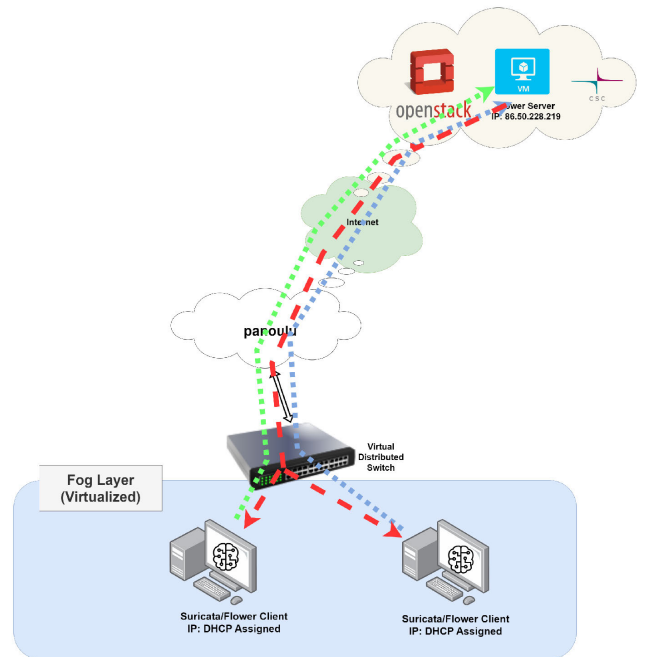


FIGURE 2. Overview of the workflow for Federated Learning in the architecture. Suricata machines in the fog layer act as FED-ML clients, autonomously collecting and storing network packets as pcap files.

Within the cloud layer, the Flower server accumulates weights from the interconnected FED-ML clients and subsequently computes their average. The updated weights are then sent back to the clients, as shown by the dashed red line in Figure 2. This entire procedure can persist for an indefinite duration as needed.

IV. IMPLEMENTATION

In this section, we turn into the core implementation aspects of SDN, NFV, and SO, shedding light on their technical underpinnings and operational synergies.

- 1) SO: In the proposed architecture, vCenter server appliance was deployed on one of the hosting ESXi² servers to allow the implementation of NFV and SDN technologies. NFV is the virtualization technique of network services to replace traditional hardware-based network functions, while SDN is an approach that separates the network control plane from the forwarding plane for the purpose of centralized network management. These technologies enable the operation of SO, as well as the monitoring of running systems, dynamic workload optimization, and proactive issue avoidance.
- 2) SDN: SDN implementation and generation of datasets for the proposed architecture is accomplished using the VMware vSphere platform, which utilizes virtual switches. Furthermore, the VMware vSphere hypervisor is deployed to create and manage multiple virtual

²ESXi version 7 was used in this work.

machines that operate concurrently to provide various services.

- 3) **NFV:** To facilitate hybrid deployments involving both physical and virtual machines across multiple domains, the VMware vCenter appliance is deployed, providing an abstracted modular design. The virtual distributed switch located within the vCenter server facilitates data transmission for both benign and malicious traffic by enabling communication between the different layers in the proposed architecture.

The fog layer incorporates the virtualized version of the suggested architecture. VMware vSphere is responsible for deploying and overseeing NFV, fog devices, and SDN components. The specific details are as follows:

- 1) **SDN management and hypervisor technology** were configured and deployed using the network and controller platforms.
 - Firstly, the vCenter server acts as a controller for the proposed architecture. It offers a server for virtualization that consolidates applications onto hardware. This centralized platform is utilized to manage vSphere systems, ensuring security by including vSphere replication and data protection features. In other words, it serves as our service orchestration (SO) tool.
 - The second component of the proposed architecture is the network, which facilitates virtualization of the testbed network and its management to enable network communications between the layers. It empowers granular overlay security and networking by offering distributed network services for Network Function Virtualizations (NFVs).
 - The proposed architecture's storage component is supported by Synology and implemented as a Storage Area Network (SAN). LUNs are provided by the SAN to vCenter to enable storage space for VMs that operate inside the ESXi hosts. The SAN and other components are co-located in the same network to minimize latency and communication jitter (Figure 1).
- 2) **NFV data center service** is facilitated by the vCenter server, where computational resources are provided by ESXi hosts for VMware vSphere. These hosts can be grouped together to provide aggregated resources called clusters.
- 3) The VMs in the fog nodes are deployed to simulate the necessary devices for generating new datasets. Five main components are deployed in the architecture for various tasks, such as executing benign and malicious scenarios and managing services between layers.

Specifically, the functionalities of these components, as illustrated in Figure 3, are detailed below.

- 1) **Regular Systems:** These systems function typically within the fog layer. They are configured to have dynamic IP addresses assigned to them within the

panOULU network. This configuration resembles a real-world scenario where all machines automatically get their IPs through a DHCP server. The regular systems have several operating systems, including Windows 7, Windows 10, Ubuntu 18.04 LTS Server, Ubuntu 20.04 LTS, and Ubuntu 14.04 LTS Server.

- 2) **Exploitable Systems:** These systems are designed with intentional vulnerabilities, making them susceptible to hacking. They mimic real-world systems that lack regular updates and possess genuine security flaws. Importantly, these systems are isolated by a firewall to prevent outbound traffic to the broader network. This measure ensures that potential attackers cannot use these exploitable systems as a gateway to compromise other parts of the network. The following lists the systems in this category:
 - **DVWA:** The Damn Vulnerable Web Application (DVWA) is built with MySQL and PHP that is deliberately created to have numerous vulnerabilities. The main goal of this penetration-testing environment is to assist security professionals and penetration testers in evaluating their capabilities and tools. Furthermore, it can assist web developers in gaining a deeper comprehension of web application security and strategies for enhancing security measures. Additionally, it can serve as a valuable educational resource for students and teachers seeking to learn about web application security and the various vulnerabilities that may arise.
 - **OWASP:** OWASP Security Shepherd is a platform designed to provide comprehensive training in the security aspects of web and mobile applications. Security Shepherd is intentionally developed to enhance and cultivate security awareness among individuals with varying levels of skills and expertise.
 - **Metasploitable3:** Metasploitable3 is a purpose-built virtual machine (VM) that is deliberately designed with an extensive array of security vulnerabilities from its inception. Its intended purpose is to serve as a designated testing target for exploring and assessing exploits using the Metasploit framework. Metasploitable3 is a freely available virtual machine that enables the simulation of attacks, primarily utilizing the capabilities of the Metasploit framework.
- 3) **Attacking Systems:** These are explicitly employed for hacking within the testbed framework. Every device displayed in the suggested cyber range architecture can be viewed as potential targets for attack scenarios. Within this context, a probing attack (scanning) targets only the devices in the fog and edge layers of the cyber range. It is crucial to restrict other attack scenarios to ensure they do not compromise all devices within the network

- 4) Collector Systems: These systems are designed to capture and aggregate network traffic throughout the entire testbed structure. For this objective, both Suricata and Wazuh were implemented. Suricata collects network traffic in the pcap format. This data is routed to Suricata both from the SPAN port of the switch and via the port forwarding capabilities of the virtual distributed switch situated at the Center. Once captured, Suricata relays these logs to Wazuh. Wazuh serves as a Security Information and Event Management (SIEM) tool, further enhancing visualization and indexing within the ELK stack.

Furthermore, the TShark tool is employed for packet-level analysis to extract valuable network packet features, which are subsequently utilized in conjunction with machine learning and deep learning algorithms. The identified patterns and anomalies in network traffic, derived from the extracted features, can be leveraged to detect potentially malicious activity within the network.

- 5) Federated Learning Systems: These systems employ the proposed architecture and infrastructure to provide the end result of recognizing malicious patterns generated by hacker machines through federated machine learning. These systems reside on the different parts of the architecture, namely the fog and cloud layers.

The Flower clients in the fog layer are deployed within the subnet and can be deployed anywhere within the geographical location covered by the panOULU network. The same principle applies to the Flower server machine, which can be deployed in any cloud computing platform, such as AWS, Azure, GCP, or privately hosted clouds. In this work, the Suricata machines, which are IDS/IPS machines, also host the Flower clients. These clients constantly read the streaming pcap files provided by Suricata and extract useful features from them through a custom-developed code.

Based on the attackers' IP addresses and the timestamp of the attacks, these packets are labelled as either benign or attack type (probe packets). After labelling, the IP addresses are stripped from the data to represent the real-world situation for the ML model, ensuring that it does not learn patterns biased by IP addresses. After the ML model is trained locally in each Suricata machine, the learned weights are sent to the Flower server in the cloud layer, which is deployed on OpenStack provided by CSC.

The Flower server then aggregates and averages the weights from all the Flower clients and sends back the updated weights to each individual Flower client. In this way, the Flower clients can learn from each other as each individual client has access to specific network traffic. The explained process can continue for as long as desired, as it is configurable during the initialization phase of the whole federated machine learning process.

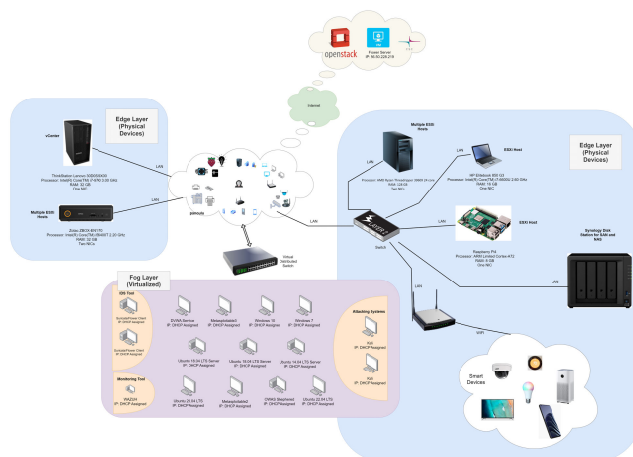


FIGURE 3. Overview of our testbed highlighting the various technology layers.

PanOULU is a municipal wireless network situated in Finland, spanning across central Oulu and encompassing the surrounding campus areas. This comprehensive network interconnects a diverse array of devices, ranging from personal computers and Internet of Things (IoT) devices to mobile phones. Within this infrastructure, the FED-ML clients are integrated into the Suricata machines. These machines not only operate as Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) components but also serve as hosts for the FED-ML clients.

In this architectural setup, the process of federated learning remains ongoing, utilizing the panOULU public network as its communication backbone. The clients are perpetually engaged in reading streaming pcap files, a format used for packet capture, which are supplied by the Suricata system. From these pcap files, the clients systematically extract valuable features. These extracted features are then utilized in classifying packets, categorizing them as either benign or indicative of specific attack types (such as probe packets). This classification is informed by the analysis of attackers' IP addresses as well as the timestamps associated with the attacks.

The IP addresses are removed from the dataset to reflect real-world conditions for the ML model, ensuring that it does not learn patterns influenced by IP addresses, thus minimizing bias. Once the ML model is trained locally on each Suricata machine, the learned weights are sent to the Flower server in the cloud layer. In this work, OpenStack provided by the CSC (also located in Finland) was chosen as the cloud computing platform to deploy the Flower server. CSC is the IT Center for Science, owned by the Finnish state and higher education institutions. Through the cloud interface, the interaction of FED-ML clients facilitates mutual learning, as each client gains insights from distinct network traffic datasets, rather than the entirety of the traffic.

V. METHODS AND APPLICATIONS

This section presents the application of the implemented architecture, emphasizing the empirical findings obtained

from the implemented FED-ML models. The models are trained using streaming data that flow through the distributed switch in different layers of the proposed architecture. The evaluation dataset, which was captured and utilized to evaluate the models' performance, has been made public in the provided link.³

A. NETWORK PERFORMANCE EVALUATION

To ensure that the network devices can support the experiment within the suggested architecture, we utilized the iperf3 tool to measure their performance. Iperf3 is a proactive measurement utility used to ascertain the upper limit of bandwidth attainable on IP networks. It provides the adjustment of multiple parameters related to timing, buffers, and protocols (including TCP, UDP, SCTP with both IPv4 and IPv6), and generates comprehensive reports for every test, encompassing details such as bandwidth, loss, and various other parameters.

The network performance evaluation was carried out in panOULU for both TCP and UDP protocols over a period of four days. The ribbon plot and box plot for the measurements are shown in Figures 4 and 5. The plots indicate that the network devices attached to the panOULU have sufficient bandwidth and speed to support the proposed architecture. This support is demonstrated via the interquartile of the box plots that represents 50 percent of the information. From the TCP box plot, we can deduce that the upload rate ranges from 820 to 830, with the whiskers representing the maximum and minimum values. Similarly, the UDP box plot suggests a rate between 260 to 270, with the whiskers indicating the range's extremes. Naturally, there are outliers influenced by network congestion, QoS configurations, bandwidth constraints during peak access times, and the number of devices or users on the network.

B. DATA COLLECTION

The initial phase involves capturing network data in pcap format to facilitate the processing of network traffic. Two of the collector systems with Suricata software installed on them were used to capture the network traffic in transit through the panOULU network. The collectors log interactions between systems within our testbed on the panOULU network, capturing their traffic for subsequent analysis. The traffic is mirrored to these systems through the SPAN port in the physical switch, enabling them to capture the entire traffic in the testbed depicted in Figure 3. The network configuration in the systems is set to promiscuous mode, which enables them to listen to the local traffic.

The captured traffic is saved as 3,000 pcap files, each with a size of 100MB, in the highest possible compressed form (level 16 of lz4 algorithm). These files are rotated upon reaching the maximum file count, with newer files replacing older ones in the sequence. It is crucial to highlight that the captured traffic is confined to the visibility scope of the virtual

machines (VMs) on which the capture software is installed. This traffic is being captured legitimately and pertains only to traffic going to and from the illustrated systems.

1) STREAMING DATA

As depicted in Figure 3, the two machines equipped with Suricata operate as Flower client machines, capturing network activities and serving as IDS systems. The data stream used in the FED-ML algorithms is sourced independently from these machines. This allows us to scrutinize the logs from these machines and provide statistical insights regarding the base data used in the initial phase of the machine learning algorithms.

The training of the selected four machine learning models on the first Suricata machine spanned a duration of 12 days. During this time, 13,488 pcap files were processed, but only 3,266 of them were used for training the models. The remaining files were excluded as they lacked any traces of the attacker's machine packets. The models underwent training using a grand total of 485,924,228 packets.

It is worth emphasizing that when dealing with streaming data, there may be pcap files that only contain regular packets and no malicious packets. If no malicious packets are included in the training data, the machine learning algorithms return an error as they expect at least two different categories. Therefore, these files were discarded during the training phase. This also underscores the importance of ML algorithms to be able to learn from data with only one class (i.e. a unary category of data).

On the second Suricata machine, the models were trained over a period of 12 days. During this period, 18,092 pcap files were processed, of which only 3,238 were used for training the models. The remaining files were discarded for the same reason discussed earlier, as they did not contain any malicious packets. A sum of 503,236,492 packets was extracted from the 3,238 pcap files in total. The percentage of regular entries during streaming was 99.63%, whereas the percentage of probe entries was 0.36%.

C. FEATURE EXTRACTION

The feature extraction process from binary pcap files was accomplished using the TShark tool. TShark is a terminal-based version of Wireshark created to capture and view packets in scenarios where an interactive user interface is not required. Using TShark, 223,474 features could be produced from each pcap file, but not all of them were applicable to the federated machine learning experiment. To narrow down the features to more usable ones, the complete set of 223,474 features was initially extracted from the pcap files. Using Python, the features with fewer null values were analyzed.

A total of 150 features with fewer null values were analyzed, and 40 features were chosen for the extraction of usable fields from pcap files during the federated machine learning phase. The selection was based on the features' usefulness and the available data for each feature.

³<https://doi.org/10.5281/zenodo.7956304>

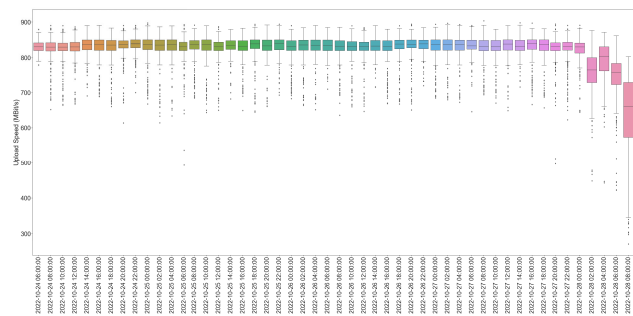


FIGURE 4. TCP box plot.

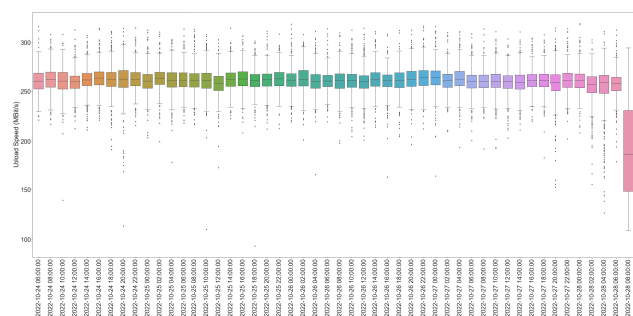


FIGURE 5. UDP box plot.

During the FED-ML phase, all categorical features were excluded from this list. The exclusion was necessary because not all data components are known during the FED-ML process, as the data is being streamed. As a result, label encoding or one-hot encoding cannot be used for categorical features.

The learning phase cycles utilized 37 features presented in the appendix section table 2. After labeling the batch data, three features are excluded: source and destination IPs, and the epoch time. Hence, the overall feature count is subsequently reduced from 40 to 37. While these features are primarily numerical, they can sometimes include categorical values based on the output from the TShark tool. In such cases, a default numerical value replaces the categorical value.

D. LABELING THE STREAMING DATA

To distinguish between normal network activities and malicious ones in the streaming data, a probing scan is performed on the victim virtual machines (VMs) within the panOULU network. The scan is performed using the attack machines that are hosted on ESXi systems under the control of the vCenter server. Criteria such as the time of scanning and the IP addresses of the hacking systems are recorded during the scan. These criteria are then used to label the streaming data.

Packets that have the attacker’s IP address in either the source or destination and include the time of the attack are categorized as attack packets. In contrast, packets not meeting these criteria are designated as normal network packets. This

classification procedure is essential for machine learning algorithms, which depend on accurately labelled packets to perform classification tasks.

Python is utilized to process the network pcap files that are supplied by the Suricata machines. The packets are labelled based on their features, which are extracted and converted to CSV format. The entire process described is fully automated, eliminating the need for any human intervention.

E. DATA ANALYTICS

The analysis of the streaming data involved utilizing the Python scripting language, coupled with various packages for implementing machine learning and deep learning techniques. Additionally, essential packages for data processing and manipulation were employed. Among the packages utilized for the analysis are scikit-learn, Pandas, numpy, Flower, and matplotlib.

During each learning round in FED-ML, the prepared dataset is partitioned into two segments, namely training and testing. The ratio of test to train is set at 1 to 4. The train portion is used to feed the FED-ML algorithms to acquire knowledge about the behaviour of the network from the streaming data. The test portion of the data is used to evaluate the performance of the model following each training iteration.

The model’s performance is logged and displayed on the terminal during the learning phase. Following every training iteration, the model’s performance is assessed by employing the test data, and the resulting data is logged to track the model’s progress. This evaluation process is crucial for determining the model’s efficacy and identifying areas that need improvement.

The trained models were saved on the hard disk for future evaluation of unseen data once the desired training rounds were satisfied. As previously mentioned, a set of 3,000 out-of-sample pcap files was employed to assess their performance. Nevertheless, processing the entire dataset simultaneously proved unfeasible due to its considerable volume. Even with a robust laboratory server with 128GB of RAM, innovative programming strategies had to be developed to surmount this obstacle.

To cope with the substantial volume of data, 110GB of RAM and 350GB of swap memory were allocated. Multi-processing techniques were implemented to expedite the entire process, allowing multiple processor cores to be utilized simultaneously. After extracting packets from these 3,000 pcap files, a total of 2,789,874,382 entries were obtained. Of this total, 2,783,207,918 packets were regular packets, while the remaining 6,666,464 were probe packets.

F. FEDERATED MACHINE LEARNING CLASSIFICATION

In the experiment, four machine learning models were established for training. After the FED-ML models completed the desired number of learning phases, they were saved in a storage device (i.e., Synology) for future classification purposes. During the FED-ML process, several commonly

used machine learning algorithms were utilized. The efficacy and efficiency of these algorithms in classifying malicious and benign packets on previously unseen data were also evaluated in this study.

Two rounds of training were conducted for the FED-ML models during the experiment. In the initial round, four models were trained in a federated manner for five rounds, and their performance was evaluated. Since no comparable experiments in the literature were similar to this study, a second round of training was also conducted. In the second round, the same four models were trained on streaming data with 50 training rounds to allow for comparisons and to identify any similarities or differences that arose due to the fluctuations in the number of training iterations.

The second training round was conducted to compare the results and identify any similarities or differences that may have arisen due to the varying number of training rounds. This comparison was intended to improve our understanding of the models' performance and their capacity to classify malicious and benign packets on unseen data. An overview of the four selected models is provided in the following.

1) LOGISTIC REGRESSION

Logistic regression is a popular statistical method used for classification, determining the likelihood of events based on predictor variables. This supervised machine learning model employs the logistic function to convert odds ratios into probabilities. It is commonly applied in areas such as fraud detection and marketing. Multinomial logistic regression extends this model, allowing it to manage multiple categories.

2) PERCEPTRON

The Perceptron is a frequently employed linear classifier in the realm of supervised learning. Its composition comprises of four fundamental elements: weights, input values, net sum, and an activation function. The input values are fed into the Perceptron and multiplied by their corresponding weights. These weighted inputs are summed up to produce a net sum, and subsequently transmitted through an activation function. The Perceptron is particularly well-suited for binary classification tasks.

3) PASSIVE AGGRESSIVE CLASSIFIER

The passive aggressive classifier is an algorithm in the field of machine learning employed for the purpose of categorizing data points into two distinct groups. Typically, these groups are the positive and negative classes. This particular algorithm shares its foundation with the Perceptron algorithm, with the addition of a regularization parameter responsible for managing the balance between expanding the margin size (In other words, it controls the proximity between the decision boundary and the nearest data points) and guarantying that the classifier avoids incorrect classification of training examples. Moreover, passive aggressive classifiers update their model selectively, solely when an error occurs, rather than after

processing each instance (i.e., every training example), which can lead to faster convergence.

4) STOCHASTIC GRADIENT DESCENT CLASSIFIER

Stochastic Gradient Descent (SGD) is an iterative optimization technique employed to minimize a cost function and determine the optimal values of function parameters/coefficients. It is a simple and efficient method utilized to train linear classifiers by minimizing convex loss functions, including SVM and Logistic regression, in the context of discriminative learning. The SGD classifier is capable of implementing a plain SGD learning routine allowing for the use of different loss functions and penalties for classification tasks

VI. RESULTS AND DISCUSSION

Table 1 details the performance metrics - Accuracy, Precision, Recall, and F1-Score - for the selected trained models, including Logistic Regression, Perceptron, Passive Aggressive, and Stochastic Gradient Descent (SGD). The metrics are delineated for both 5 and 50 rounds of training, derived from the evaluation on 3000 pcap files as referenced earlier. A cursory glance suggests that post 50 training rounds, models achieve high accuracy levels. However, the associated metrics, notably precision, recall, and F1-score, yield underwhelming results, predominantly falling below the 50% mark. Delving deeper into the comparative analysis, performance metrics show a noticeable upward trend as the training rounds are augmented to 50 for all models. The Logistic Regression model is a testament to this, registering a commendable 99.74% accuracy in its 50-round training. Yet, its F1-score of zero raises concerns of possible overfitting or challenges associated with class imbalance. In contrast, the Perceptron and Passive Aggressive models exhibit a balanced performance between precision and recall after 50 training rounds, highlighting their potential for real-world applications and further optimization.

The development of AI-driven intrusion detection systems necessitates using high-quality data for model training. Over the years, numerous datasets have been made publicly available to assist researchers in developing high-performance models. However, many of the previously introduced architectures and datasets have become outdated. As such, this study primarily aims to design a contemporary testbed framework apt for defense mechanisms, considering the continuous advancement of technologies and emerging Federated Learning models. Subsequent sections delve into the advantages and drawbacks of the suggested approach.

The developed testbed is unique compared to other testbeds discussed in the prior research section (II) because we utilize a public network for our experiments, while most existing testbeds rely on private networks. This difference brings several strengths that distinguish our approach from other testbeds. One of the distinctive features of this work is the streaming network data that is fed into FED-ML models. The data streaming is subjected to preprocessing and labelled

TABLE 1. Performance metrics of the trained models.

Model	Training Round	Accuracy	Precision	Recall	F1-Score
Logistic Regression	5	0.3398	0.0011	0.2248	0.0021
	50	0.9974	0.0045	0.0000	0.0000
Perceptron	5	0.3684	0.0014	0.4291	0.0028
	50	0.9870	0.0889	0.2406	0.0897
Passive Aggressive	5	0.3224	0.0024	0.6172	0.0047
	50	0.9953	0.1115	0.1728	0.1005
Stochastic Gradient Descent	5	0.3616	0.0000	0.0353	0.0000
	50	0.9902	0.0460	0.1189	0.1189

as they flow in the network, enabling data collection and pre-processing to occur concurrently with the learning phase of FED-ML models.

Diverging from the conventional literature, which often adheres to a sequential method of data collection followed by preprocessing for ML models, our methodology introduces an innovative approach. Unlike the conventional process that lacks concurrency and necessitates manual human intervention at multiple interruptions in the pipeline, our approach empowers the models in use to capture the dynamic nature of network flow in real-time.

Nevertheless, the experimental results attained in this study were less than optimal, providing an opportunity for further discussion and analysis. Several related studies have employed IP addresses for classification purposes. However, this approach presents various potential shortcomings. Hackers frequently alter their IP addresses deliberately to evade detection, and machine learning algorithms excel in discerning patterns within datasets, IP addresses included. If a model is trained to link malevolent activities to specific attacker IPs, its efficacy may be compromised in real-world scenarios where IP addresses are regularly changing and dynamic. While IP addresses were not employed during the learning phase of this work, their inclusion remains a subject of discussion. Using IP addresses can potentially introduce biased results, and disregarding them can lead to inaccurate outcomes.

It is worth emphasizing that the models employed in this study have been trained using streaming data, meaning they have seen different data streams during their training phase. This approach is nonconformist compared to traditional machine learning models examined in the earlier research section, which are often trained using the same data set. Nonetheless, given that the models in this study are exposed to a substantial amount of data during their learning phase, minor variations in their learning data are unlikely to significantly impact over an extended period.

To ensure fair comparisons between the different models, they are all evaluated using a consistent set of test data, even though they were trained on different data streams. This approach guarantees that the evaluation metrics are reported against the same set of data, allowing us to make accurate comparisons between the models. This model evaluation method differs from other studies in the field, where the models are also assessed using the same set of test data.

Normalization was not employed as part of the learning procedure in this study. While some algorithms benefit from normalization and standardization [9], it was not possible to perform this step in the current study due to the streaming nature of the data. Previous works, such as [28], [33], and [58], have utilized normalization techniques in their work. However, in the proposed architecture of this study, each round during the learning phase could be considered as a new opportunity for the machine learning models to be appropriately trained, and normalization could still be applied.

Feature selection has not been fully optimized, leaving room for the possibility of other features that could offer advantages to the models. However, it should be highlighted that string features were ineligible for selection since machine learning models only recognize numbers. While it is possible to transform strings into numbers through the process of label encoding or one-hot encoding, this presents another problem in the case of streaming data processing. In contrast to the conventional approach adopted by other relevant studies, where all the data is accessible during the learning phase, streaming data is not instantly available in its entirety.

The rationale behind refraining from transforming categorical data into numerical equivalents in real-time streaming is due to the potential emergence of new categories within future data. Converting categorical values into numerical equivalents is common in related works, such as [37]. Nonetheless, this demand presents a notable challenge for the context of federated learning, particularly in the realm of streaming data. A key discovery of this study is the imperative to tackle this challenge effectively.

There was a notable disparity between the quantity of regular packets and probe packets in this work, which may have had adverse effects on the efficacy of machine learning algorithms trained on this data. A balanced dataset is crucial for optimal model performance, representing each category proportionally. In our case, this would require a more balanced representation of regular and probe packets. To address this issue, we could increase the number of probe packets by conducting reconnaissance and gathering more data from the network. One way to do this is by scaling out the number of hacker machines, which could increase the number of probe packets generated. However, increasing the number of hacker machines must be done carefully and ethically, with proper precautions taken to prevent any harm to the

panOULU network or its users. Other related studies such as [25], [26], and [29] have addressed imbalance by resampling data from the same dataset. However, this approach can lead to overfitting, where the model exhibits excessive fitting to the training data and performs poorly on new data.

Using packet-level features alone in this work may not be sufficient for accurately identifying malicious activities in network traffic data. To extract features at the packet-level, TShark was utilized to process pcap files. However, this approach did not consider the sequences of back-and-forth packets, as this could be quite challenging due to the complexity of the architecture with devices connecting from different layers in the cyber range. On the other hand, several related studies mentioned in the prior research section have utilized common benchmark datasets that are available for intrusion detection research. These datasets typically consist of various features generated using different tools and scripts. Utilizing these datasets with distinct features could positively impact the results of these studies. For example, this work excluded IP addresses of the packets during the learning phase of the models, whereas in the study by Moustafa [39], these features were included.

A. REVISITING THE RESEARCH CHALLENGES

In the introduction section, we established four essential research questions that serve as the foundation for this study. Subsequently, we will provide explicit answers to each of these questions as follows:

Challenge 1: The virtualization of network functions and the utilization of software-defined networking provided by VMware vSphere enabled the deployment of the Distributed Switch in the proposed architecture, allowing all devices in the fog layer to communicate with each other, as well as with the edge layer and cloud layer. This approach facilitated the creation of a unified virtual switch that spans across multiple physical hosts, thereby enabling centralized network management and configuration. This infrastructure was subsequently harnessed for the capture and analysis of PCAP files.

Furthermore, the service orchestration feature provided by the vCenter Server Appliance enabled integrated control of the devices in the fog layer. This feature streamlined the process of deploying, managing, and monitoring virtual machines in the testbed.

Furthermore, the federated learning architecture employed in this study facilitated the integration of edge and cloud computing. The devices equipped with Suricata played a pivotal role by training the models locally and transmitting the updated outcomes to the cloud. The Flower server acted as the platform for Flower clients in the cloud layer. This trait of being locally trained had the benefit of reducing latency and providing a cost-effective way of managing large volumes of data.

Challenge 2: Federated learning has the distinctive feature of preserving privacy at the edge layer without revealing any information about the data. This paradigm was employed in

this work to address security concerns. Specifically, this was achieved by training models locally with the streaming data and updating the Flower server with the trained models.

This paradigm offers the potential to expand the utilization of federation to various layers in alternative scenarios. This level of granularity implies that even edge devices, such as Raspberry Pi, could independently train models on a local scale without requiring access to the complete network traffic. They could effectively train the models on their local data and subsequently push updates back to the Flower server.

Challenge 3: The proposed architecture aimed to showcase the intricate nature of modern-day network topologies. The setup includes wired and wireless connections to the panOULU network, encompassing diverse topologies. The devices utilized in this study encompass a range of technologies, including IoT devices, smartphones, virtual machines (VMs), and others.

A realistic testbed has been established by combining these devices with various communication media and technologies. This testbed includes different tiers, such as edge, fog, and cloud, which can be utilized to generate a high-quality and realistic dataset. The inclusion of diverse devices and topologies in this architecture emphasizes the intricacy of contemporary networks. With the increasing number of IoT devices and the need for seamless connectivity, developing such a testbed that accurately replicates real-world scenarios has become crucial.

Challenge 4: Based on the findings of this study, it can be concluded that securing devices in a public network can be a challenging task. The table 1 reporting the metrics of the trained machine learning models demonstrates the difficulty of accurately predicting malicious activities. Despite extensive research in this field, the results have not been entirely satisfactory. Many models are based on outdated datasets or datasets created on private networks, rendering them less effective when applied to public networks. Additionally, most models reported by other studies are not trained using streaming data, which can be ineffective in the real world. It could be inferred from the results that the best way to protect network assets and devices is by securing them in the first place by setting up strict firewall rules and limiting these devices' inbound and outbound activities. Keeping the operating systems of these devices up-to-date is another precaution that could mitigate the security breach of these devices.

B. LIMITATIONS AND FUTURE WORK

A selection of ML algorithms was chosen to evaluate the efficacy of a novel testbed. It is possible to further expand this selection to include other ML algorithms with different methodologies. Additionally, one FED-ML master node and two slave nodes were used to conduct the experiment. In future studies, it may be feasible to scale out the number of slave nodes. However, such scaling may result in a substantial rise in the ratio of probe packets to regular packets, ultimately

TABLE 2. The 37 features employed during FED-ML phase along with example values.

Feature	Description	Example Value
arp.hw.type	Hardware type	1
arp.hw.size	Hardware size	6
arp.proto.size	Protocol size	4
arp.opcode	Opcode	2
data.len	Length	2713
eth.dst.lg	LG bit	1
eth.dst.ig	IG bit	1
eth.src.lg	LG bit	1
eth.src.ig	IG bit	1
frame.offset_shift	Time shift for this packet	0
frame.len	Frame length on the wire	1208
frame.cap_len	Frame length stored into the capture file	215
frame.marked	Frame is marked	0
frame.ignored	Frame is ignored	0
frame.encap_type	Encapsulation type	1
gre	Generic Routing Encapsulation	'Generic Routing Encapsulation (IP)'
ip.version	Version	6
ip.hdr_len	Header Length	24
ip.dsfield.dscp	Differentiated Services Codepoint	56
ip.dsfield.ecn	Explicit Congestion Notification	2
ip.len	Total Length	614
ip.flags.rb	Reserved bit	0
ip.flags.df	Don't fragment	1
ip.flags.mf	More fragments	0
ip.frag_offset	Fragment Offset	0
ip.ttl	Time to Live	31
ip.proto	Protocol	47
ip.checksum.status	Header checksum status	2
tcp.srcport	Source Port	53425
tcp.flags	Flags	0x00000098
tcp.flags.ns	Nonce	0
tcp.flags.cwr	Congestion Window Reduced (CWR)	1
udp.srcport	Source Port	64413
udp.dstport	Destination Port	54087
udp.stream	Stream index	1345
udp.length	Length	225
udp.checksum.status	Checksum Status	3

resulting in a more balanced dataset compared to the current status of the data ratio.

In the future, we plan to extend this work by performing different types of attacks beyond just probing the cyber range and training ML models on these diverse attacks. The ML models utilized in this work can predict multi-class data, making it easier to expand this study to multi-attack-type scenarios for FED-ML learning. Moreover, deploying multiple honeypots in various ESXi hosts in a distributed manner is possible to attract attackers engaging in malicious activities within the panOULU network. The resulting data can be utilized to train ML algorithms for cybersecurity purposes. The proposed architecture can provide the necessary platform for implementing the described blueprint.

VII. CONCLUSION

In conclusion, our research harnessed the capabilities of VMware vSphere to develop a Distributed Switch, promoting seamless communication across fog, edge, and cloud layers.

The centralized network management system offered an efficient way to analyze PCAP files, and service orchestration streamlined the deployment, management, and monitoring of virtual machines.

The introduction of federated learning ensured data privacy at the edge layer. By training models locally and updating only the model information to the Flower server, we managed to reduce both latency and the costs associated with large data volumes, all while preserving privacy. This approach has the potential to revolutionize how we handle data, especially in edge devices like Raspberry Pi.

Our study also shed light on the complexity of current network topologies. By establishing a testbed that integrated various communication connections, devices, and technologies, we aimed to offer a realistic representation of the intricacies found in contemporary networks. As the world continues to become more connected, and with the proliferation of IoT devices, understanding and replicating these complexities is crucial.

Lastly, our findings emphasized the inherent challenges in securing devices on public networks. Despite the vast amount of research dedicated to predicting malicious activities using machine learning models, many fall short, often due to outdated datasets. The results suggest that preventive measures, such as establishing strict firewall protocols and ensuring regular updates to device operating systems, are pivotal in mitigating potential security breaches.

APPENDIX TABLES

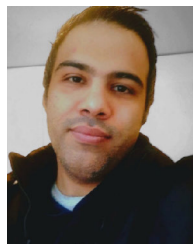
See Table 2.

REFERENCES

- [1] *ADFA Intrusion Detection Datasets*. Accessed: Jun. 29, 2023. [Online]. Available: <https://research.unsw.edu.au/projects/adfa-ids-datasets>
- [2] *CDX Dataset*. Accessed: Jun. 29, 2023. [Online]. Available: <http://www.fit.vutbr.cz/~ihomoliak/asnm/ASN-CDX-2009.html>
- [3] *Center for Applied Internet Data Analysis*. Accessed: Jun. 29, 2023. [Online]. Available: <https://www.caida.org/data/overview/>
- [4] *CIC DoS Dataset*. Accessed: Jun. 29, 2023. [Online]. Available: <https://www.umb.ca/cic/datasets/dos-dataset.html>
- [5] *ISCX Dataset*. Accessed: Jun. 29, 2023. [Online]. Available: <https://www.umb.ca/cic/datasets/ids.html>
- [6] *MAWI Datasets*. Accessed: Jun. 29, 2023. [Online]. Available: <http://www.fukuda-lab.org/mawilab/data.html>
- [7] R. C. Agarwal and M. V. Joshi, "PNrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection)," in *Proc. SIAM Int. Conf. Data Mining (SDM)*, 2001, pp. 1–17.
- [8] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Comput. Commun.*, vol. 195, pp. 346–361, Nov. 2022.
- [9] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, Jan. 2021, Art. no. e4150.
- [10] V. Alcácer and V. Cruz-Machado, "Scanning the Industry 4.0: A literature review on technologies for manufacturing systems," *Eng. Sci. Technol., Int. J.*, vol. 22, no. 3, pp. 899–919, Jun. 2019.
- [11] I. Almomani, B. Al-Kasasbeh, and M. AL-Akhras, "WSN-DS: A dataset for intrusion detection systems in wireless sensor networks," *J. Sensors*, vol. 2016, Sep. 2016, Art. no. 4731953.

- [12] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 195–200.
- [13] S. Althubiti, W. Nick, J. Mason, X. Yuan, and A. Esterline, "Applying long short-term memory recurrent neural network for intrusion detection," in *Proc. SoutheastCon*, Apr. 2018, pp. 1–5.
- [14] Z. Amiri, A. Heidari, N. J. Navimipour, and M. Unal, "Resilient and dependability management in distributed environments: A systematic and comprehensive literature review," *Cluster Comput.*, vol. 26, no. 2, pp. 1565–1600, Apr. 2023.
- [15] S. Basumallik, R. Ma, and S. Eftekharijad, "Packet-data anomaly detection in PMU-based state estimator using convolutional neural network," *Int. J. Electr. Power Energy Syst.*, vol. 107, pp. 690–702, May 2019.
- [16] M. Botha, R. Von Solms, K. Perry, E. Loubser, and G. Yamony, "The utilization of artificial intelligence in a hybrid intrusion detection system," in *Proc. Annu. Res. Conf. South Afr. Inst. Comput. Scientists Inf. Technologists Enablement Through Technol.*, 2002, pp. 149–155.
- [17] L. Cavallaro, J. Kinder, F. Pendlebury, and F. Pierazzi, "Are machine learning models for malware detection ready for prime time?" *IEEE Secur. Privacy*, vol. 21, no. 2, pp. 53–56, Mar. 2023.
- [18] C. G. Cordero, S. Hauke, M. Mühlhäuser, and M. Fischer, "Analyzing flow-based anomaly intrusion detection using replicator neural networks," in *Proc. 14th Annu. Conf. Privacy, Secur. Trust (PST)*, Dec. 2016, pp. 317–324.
- [19] F. Feng, X. Liu, B. Yong, R. Zhou, and Q. Zhou, "Anomaly detection in ad-hoc networks based on deep learning model: A plug and play device," *Ad Hoc Netw.*, vol. 84, pp. 82–89, Mar. 2019.
- [20] M. A. Ferrag, L. Maglaras, S. Moschogiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419.
- [21] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *Proc. Int. Conf. Inf. Sci. Secur. (ICISS)*, Dec. 2016, pp. 1–6.
- [22] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.
- [23] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BIONETICS)*, May 2016, pp. 21–26.
- [24] F. Jiang, Y. Fu, B. B. Gupta, Y. Liang, S. Rho, F. Lou, F. Meng, and Z. Tian, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 2, pp. 204–212, Apr. 2020.
- [25] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access*, vol. 8, pp. 32464–32476, 2020.
- [26] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020.
- [27] S. M. Kasongo and Y. Sun, "A deep learning method with filter based feature engineering for wireless intrusion detection system," *IEEE Access*, vol. 7, pp. 38597–38607, 2019.
- [28] V. Kelli, V. Argyriou, T. Lagkas, G. Fragulis, E. Grigoriou, and P. Sarigiannidis, "IDS for industrial applications: A federated learning approach with active personalization," *Sensors*, vol. 21, no. 20, p. 6743, Oct. 2021.
- [29] F. A. Khan, A. Gumaie, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.
- [30] J. Kim, J. Kim, H. L. Thi Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Feb. 2016, pp. 1–5.
- [31] C.-S. Li and W. Liao, "Software defined networks," *IEEE Commun. Mag.*, vol. 51, no. 2, p. 113, Feb. 2013.
- [32] D. Li, L. Deng, M. Lee, and H. Wang, "IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning," *Int. J. Inf. Manage.*, vol. 49, pp. 533–545, Dec. 2019.
- [33] S. Li, Q. Qi, J. Wang, H. Sun, Y. Li, and F. R. Yu, "GGs: General gradient sparsification for federated learning in edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.
- [34] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proc. DARPA Inf. Survivability Conf. Expo. (DISCEX)*, Jan. 2000, pp. 12–26.
- [35] L. A. Maglaras, K.-H. Kim, H. Janicke, M. A. Ferrag, S. Rallis, P. Fragkou, A. Maglaras, and T. J. Cruz, "Cyber security of critical infrastructures," *ICT Exp.*, vol. 4, no. 1, pp. 42–45, Mar. 2018.
- [36] A. B. Z. Mahmoodi, "Federated learning for distributed intrusion detection systems in public networks," M.S. thesis, Fac. Inf. Technol. Elect. Eng., Univ. Oulu, Oulu, Finland, 2023.
- [37] P. H. Mirzaee, M. Shojafar, Z. Pooranian, P. Asefi, H. Cruickshank, and R. Tafazolli, "FIDS: A federated intrusion detection system for 5G smart metering network," in *Proc. 17th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2021, pp. 215–222.
- [38] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 686–728, 1st Quart., 2019.
- [39] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustain. Cities Soc.*, vol. 72, Sep. 2021, Art. no. 102994.
- [40] N. Moustafa, M. Ahmed, and S. Ahmed, "Data analytics-enabled intrusion detection: Evaluations of ToN_IoT Linux Datasets," 2020, *arXiv:2010.08521*.
- [41] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
- [42] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Netw.*, vol. 8, no. 3, pp. 26–41, May/June 1994.
- [43] J. O. Nehinbe, "A critical evaluation of datasets for investigating IDSs and IPSs researches," in *Proc. IEEE 10th Int. Conf. Cybernetic Intell. Syst. (CIS)*, Sep. 2011, pp. 92–97.
- [44] J. O. Nehinbe, "A simple method for improving intrusion detections in corporate networks," in *Information Security and Digital Forensics*, D. Weerasinghe, Ed. Berlin, Germany: Springer, 2010, pp. 111–122.
- [45] P. O'Donovan, C. Gallagher, K. Bruton, and D. T. J. O'Sullivan, "A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications," *Manuf. Lett.*, vol. 15, pp. 139–142, Jan. 2018.
- [46] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, Dec. 1999.
- [47] E. Peltonen, M. Bennis, M. Capobianco, M. Debbah, A. Ding, F. Gil-Castiñeira, M. Jurmu, T. Karvonen, M. Kelanti, A. Kliks, T. Leppänen, L. Lovén, T. Mikkonen, A. Rao, S. Samarakoon, K. Seppänen, P. Sroka, S. Tarkoma, and T. Yang, "6G white paper on edge intelligence," 2020, *arXiv:2004.14850*.
- [48] A. Rajagopal, M. Balmakhtar, and L. W. Paczkowski, "Network function virtualization (NFV) software-defined network (SDN) network-to-network interfaces (NNIs)," U.S. Patent 10 164 914, Dec. 25, 2018.
- [49] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme, in *Soft Computing in Industrial Applications*, A. Gaspar-Cunha, R. Takahashi, G. Schaefer, and L. Costa, Eds. Berlin, Germany: Springer, 2011, pp. 293–303.
- [50] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*. Setúbal, Portugal: SciTePress, 2018, pp. 108–116.
- [51] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
- [52] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," presented at the 1st Workshop Building Anal. Datasets Gathering Exper. Returns Secur., Salzburg, Austria, 2011.
- [53] T. Taleb, C. Benzaïd, M. B. Lopez, K. Mikhaylov, S. Tarkoma, P. Kostakos, N. Huda Mahmood, P. Pirinen, M. Matinmikko-Blue, M. Latva-aho, and A. Pouttu, "6G system architecture: A service of services vision," *ITU J. Future Evolving Technol.*, vol. 3, no. 3, pp. 710–743, 2022.
- [54] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.

- [55] L. J. Trautman and P. C. Ormerod, "Corporate directors' and officers' cybersecurity standard of care: The Yahoo data breach," *Amer. Univ. Law Rev.*, vol. 66, p. 1231, 2016.
- [56] A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," in *Proc. 3rd Int. Workshop Recent Adv. Intrusion Detection (RAID)*. Toulouse, France: Springer, Oct. 2000, pp. 80–93.
- [57] S. Venkatraman and M. Alazab, "Use of data visualisation for zero-day malware detection," *Secur. Commun. Netw.*, vol. 2018, Dec. 2018, Art. no. 1728303.
- [58] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
- [59] S. Zahoor, I. Ahmad, M. T. B. Othman, A. Mamoon, A. U. Rehman, M. Shafiq, and H. Hamam, "Comprehensive analysis of network slicing for the developing commercial needs and networking challenges," *Sensors*, vol. 22, no. 17, p. 6623, Sep. 2022.
- [60] Y. Zhang, P. Li, and X. Wang, "Intrusion detection for IoT based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31711–31722, 2019.
- [61] A. Özgür and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ Preprints*, vol. 4, pp. 1–21, Apr. 2016, Art. no. e1954v1.



SAEID SHEIKHI (Member, IEEE) received the bachelor's and master's degrees in software engineering. He is currently pursuing the Ph.D. degree in computer science and engineering with the University of Oulu, Finland. He is also a researcher. He has authored multiple research papers and actively participated in conferences within his field. His extensive research endeavors encompass a range of subjects, including cybersecurity, machine learning, and the security aspects of emerging networks like 5G and 6G communication systems.



ELLA PELTONEN (Senior Member, IEEE) received the Ph.D. degree from the University of Helsinki, Finland, and the Postdoctoral degree from University College Cork, Ireland. She is currently an Assistant Professor (tenure track) with the University of Oulu, Finland. Her research interests include pervasive intelligent systems and services in the 5G/6G era.



ALIREZA BAKHSHI ZADI MAHMOODI received the master's degree from the University of Oulu, Finland, where he is currently pursuing the Ph.D. degree under the guidance of Dr. Ella Peltonen. His research interests include the intersection of vehicular computing, cybersecurity, and distributed systems.



PANOS KOSTAKOS is currently a Senior Research Fellow with the Center for Ubiquitous Computing, University of Oulu. His research interests include the intersection of AI, information security, and security orchestration, focusing on autonomous, mutable, and cognitive cyber defence mechanisms. He leads the research group Cyber Security Informatics (CSI).

...