FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Alireza Bakhshi Zadi Mahmoodi

FEDERATED LEARNING FOR DISTRIBUTED INTRUSION DETECTION SYSTEMS IN PUBLIC NETWORKS

Master's Thesis
Degree Programme in Computer Science and Engineering
May 2023

Bakhshi Zadi Mahmoodi A. (2023) Federated Learning for Distributed Intrusion Detection Systems in Public Networks. University of Oulu, Degree Programme in Computer Science and Engineering, 71 p.

ABSTRACT

The rapid integration of technologies such as IoT devices, cloud, and edge computing has led to a progressively interconnected network of intelligent environments, services, and public infrastructures. This evolution highlights the critical need for sophisticated and self-governing Intrusion Detection Systems (IDS) to enhance trust and ensure the security and integrity of these interconnected environments. Furthermore, the advancement of AI-based Intrusion Detection Systems hinges on the effective utilization of high-quality data for model training. A considerable number of datasets created in controlled lab environments have recently been released, which has significantly facilitated researchers in developing and evaluating resilient Machine Learning models. However, a substantial portion of the architectures and datasets available are now considered outdated. As a result, the principal aim of this thesis is to contribute to the enhancement of knowledge concerning the creation of contemporary testbed architectures specifically designed for defense systems. The main objective of this study is to propose an innovative testbed infrastructure design, capitalizing on the broad connectivity panOULU public network, to facilitate the analysis and evaluation of AI-based security applications within a public network setting. The testbed incorporates a variety of distributed computing paradigms including edge, fog, and cloud computing. It simplifies the adoption of technologies like Software-Defined Networking, Network Function Virtualization, and Service Orchestration by leveraging the capabilities of the VMware vSphere platform. In the learning phase, a custom-developed application uses information from the attackers to automatically classify incoming data as either normal or malicious. This labeled data is then used for training machine learning models within a federated learning framework (FED-ML). The trained models are validated using previously unseen network data (test data). The entire procedure, from collecting network traffic to labeling data, and from training models within the federated architecture, operates autonomously, removing the necessity for human involvement. The development and implementation of FED-ML models in this thesis may contribute towards laying the groundwork for future-forward, AI-oriented cybersecurity measures. The dataset and testbed configuration showcased in this research could improve our understanding of the challenges associated with safeguarding public networks, especially those with heterogeneous environments comprising various technologies.

Keywords: Network Security, Cybersecurity, Federated Learning, Data Engineering, Distributed Computing, Stream Processing

Bakhshi Zadi Mahmoodi A. (2023) Diplomityön teko-ohjeet. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 71 s.

TIIVISTELMÄ

Teknologioiden, kuten IoT-laitteiden, pilvipalveluiden ja reunalaskennan, nopea integraatio on johtanut älykkäiden ympäristöjen, palveluiden ja julkisten infrastruktuurien asteittain yhteenliittyneeseen verkkoon. Tämä kehitys korostaa edistyneiden ja itsehallinnollisten tunkeutumisen havaitsemisjärjestelmien (IDS) kriittistä tarvetta parantaa luottamusta ja varmistaa näiden toisiinsa yhdistettyjen ympäristöjen turvallisuus ja eheys. Lisäksi tekoälypohjaisten tunkeutumisen havaitsemisjärjestelmien kehitys riippuu korkealaatuisen datan tehokkaasta hyödyntämisestä mallikoulutuksessa. Viime aikoina on julkaistu huomattava määrä kontrolloiduissa laboratorioympäristöissä luotuja tietojoukkoja, mikä on merkittävästi helpottanut tutkijoita kestävien koneoppimismallien kehittämisessä ja arvioinnissa. Huomattava osa käytettävissä olevista arkkitehtuureista ja tietojoukoista katsotaan kuitenkin nyt vanhentuneiksi. Tästä johtuen tämän opinnäytetyön päätavoitteena on myötävaikuttaa tietoisuuden lisäämiseen erityisesti puolustusjärjestelmiin suunniteltujen nykyaikaisten testialustojen arkkitehtuurien luomisesta. Tämän tutkimuksen päätavoitteena on ehdottaa innovatiivista testialustan infrastruktuurirakennetta, jossa hyödynnetään laajaa liitettävyyttä panOULU-julkista verkkoa, mikä helpottaa tekoälypohjaisten tietoturvasovellusten analysointia ja arviointia julkisessa verkkoympäristössä. Testbed sisältää useita hajautettuja tietojenkäsittelyparadigmoja, mukaan lukien reuna-, sumu- ja pilvilaskenta. Se yksinkertaistaa teknologioiden, kuten ohjelmiston määrittämän verkon, verkkotoimintojen virtualisoinnin ja palvelun järjestämisen, käyttöönottoa hyödyntämällä VMware vSphere -alustan ominaisuuksia. Oppimisvaiheessa räätälöity sovellus käyttää hyökkääjien tietoja luokitellakseen saapuvat tiedot automaattisesti joko normaaliksi tai haitallisiksi. Tätä merkittyä dataa käytetään sitten koneoppimismallien opetukseen liitetyssä oppimiskehyksessä (FED-ML). Koulutetut mallit validoidaan käyttämällä aiemmin näkemättömiä verkkotietoja (testitietoja). Koko prosessi verkkoliikenteen keräämisestä merkintädatan keräämiseen ja liitetyn arkkitehtuurin koulutusmalleihin toimii itsenäisesti, mikä poistaa ihmisen osallistumisen tarpeen. FED-ML-mallien kehittäminen ja käyttöönotto tässä opinnäytetyössä voi auttaa luomaan pohjaa tulevaisuuden, tekoälyyn suuntautuneille kyberturvallisuustoimenpiteille. Tässä tutkimuksessa esitellyt tietojoukot ja testialustojen konfiguraatiot voisivat parantaa ymmärrystämme julkisten verkkojen turvaamiseen liittyvistä haasteista, erityisesti sellaisista, joissa on heterogeeniset ympäristöt, jotka sisältävät erilaisia teknologioita.

Avainsanat: Verkkoturvallisuus, kyberturvallisuus, yhdistetty oppiminen, tietotekniikka, hajautettu tietojenkäsittely, suoratoiston käsittely

TABLE OF CONTENTS

FOREWORD

I am delighted to present this thesis as part of my MSc in Computer Science and Engineering, which signifies the culmination of rigorous work spanning several months. It is my honour to share my discoveries with the academic community and contribute to the collective body of knowledge in this field. I would like to extend my heartfelt gratitude to my thesis advisor, Dr. Panos Kostakos, for his invaluable assistance, backing, and motivation during the undertaking. His knowledge and input have played a vital role in influencing the direction of my research and writing. I would also like to thank the staff at the University of Oulu's Center for Ubiquitous Computing (UBICOMP) for their support during my studies. The resources and facilities provided by the Center have been invaluable in enabling me to complete this research project. Lastly, I aim for this research to add value to the existing pool of knowledge and understanding about Cybersecurity. By analyzing the performance and effectiveness of these systems, we can develop new strategies for protecting computer networks against cyber-attacks and safeguarding sensitive information. Thank you for taking the time to read this thesis, and I hope you find it informative and insightful.

Oulu, May 24th, 2023

Alireza Bakhshi Zadi Mahmoodi

LIST OF ABBREVIATIONS AND SYMBOLS

AI	Artificial Intelligence
AUC	Area Under the ROC Curve
DDoS	Distributed Denial of Service
CNIs	Critical National Infrastructures
CVE	Common Vulnerability Exposure
CNN	Convolutional Neural Network
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service
DDoS	Distributed Denial of Service
DVWA	Damn Vulnerable Web Application
FAR	False Alarm Rates
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
FFDNN	Feed Forward Deep Neural Network
FED-ML	Federated Machine Learning
HIDS	Host-based Intrusion Detection System
IoT	Internet of Things
IIoT	Industrial IoT
IT	Information Technology
ICS	Industrial Control System
IP	Internet Protocol
IaaS	Infrastructure as a Service
JS	Java Script
LEACH	Low-Energy Adaptive Clustering Hierarchy
LSTM	Long Short-Term Memory
LAN	Local Area Network
LUN	Logical Unit Number
MADMAID	Mining Audit Data for Automated Models for ID
NFV	Network Function Virtualization
NIDS	Network-based Intrusion Detection System
NIC	Network Interface Card
OT	Operational Technology
OS	Operating System
PaaS	Platform as a Service
R2L	Remote to Local
RBM	Restricted Boltzmann Machine
RFE	Recursive Feature Elimination
SDN	Software-Defined Network
SO	Service Orchestration
SCADA	Supervisory Control and Data Acquisitions
SSH	Secure Shell
SQL	Structured Query Language
SaaS	Software as a Service
SAN	Storage Area Network

TTL	Time to Live
TCP	Transmission Control Protocol
TPR	True Positive Rate
U2R	User to Root
VM	Virtual Machine
WSN	Wireless Sensor Networks

1. INTRODUCTION

As we journey into an era characterized by technological advancements, the surging growth of interconnected devices such as sensors, Internet of Things (IoT) devices, cellphones, and drones underscores the pressing need for a solid and efficient communication infrastructure. The constraints on capacity and latency inherent in the current 4G and 5G networks highlight the urgency for innovative solutions to overcome these limitations. 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. This revolution signifies a shift from the conventional "network of networks" model to a forward-thinking "service of services" paradigm [1].

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 (IoT) that's 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 technology (OT), giving rise to AI-enabled systems enhancing productivity and flexibility. Through the use of sensors, actuators, and software [2], IoT systems can be managed and controlled remotely, leading to faster manufacturing, enhanced customization, and the emergence of innovative business models [3].

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 [4], are now increasingly susceptible to cyber attacks. Recognizing this vulnerability, global powers, including the European Union, have initiated security measures and policies to safeguard these systems [5]. 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 designed to detect and prevent cyber threats.

1.1. Problem Statement

The rapid rise of Networks and Information and Communications Technology (ICT) systems have made sensitive data constantly exposed to potential attacks from both internal and external intruders [6]. High-profile security breaches like those of Bitcoin and Yahoo, which resulted in losses of millions of dollars [7], illustrate the severity of the issue. Advances in software, hardware, and network topologies, including those related to the Internet of Things (IoT), have enabled the development of increasingly sophisticated attack algorithms [8].

To address these concerns, a robust Intrusion Detection System (IDS) is needed to detect and classify attacks, intrusions, and violations of security policies promptly.

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) [9].

AI-empowered cyber solutions, specializing in threat hunting, intrusion detection, privacy protection, malware detection, and digital forensics, can be a response to some of these challenges [2]. These are centralised solutions that rely on training datasets to assess their performance and make informed decisions. To analyze network traffic flows, there are three standard methods: stateful protocol analysis, anomaly detection, and misuse detection [10]. 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 [9]. 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 [10]. 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 challenge at hand involves creating a comprehensive, economical, and adaptive security testbed characterisation. This testbed should effectively address 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. The central aim of this thesis is to tackle the stated problem, with the following research questions serving as the focus of our investigation:

- **Research Question 1:** How do key enabling technologies, such as Network Function Virtualization (NFV), Software-Defined Networking (SDN), edge/cloud computing, and federated learning (FL) help in defining novel testbed characterisations?

- **Research Question 2:** How can federated learning (FL) 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?

- **Research Question 3:** How can we set up realistic and comprehensive testbeds for evaluating proposed security techniques, given the complexity of public networks, systems, and the lack of high-quality, representative datasets?

- **Research Question 4:** What are the challenges in designing secure and distributed architectures for complex systems like IoT networks, and how can these challenges be addressed to protect critical national infrastructures from cyber attacks?

1.2. Contribution

The motivation of this work is to address the aforementioned research challenges by proposing the creation of an orchestrated testbed infrastructure for cybersecurity experimentation that includes various systems and devices on edge, fog and cloud layers that are interconnected to each other using the panOULU public network. 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) were enabled by utilizing on-premise hardware resources at the University of Oulu.

Consequently, the contribution of this work lies in presenting 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 activities, is employed to train various ML models in a federated manner. Specifically, this work employs a federated learning architecture to enhance cybersecurity in the context of intrusion detection systems (IDS) by leveraging a proposed testbed augmented with custom Python code. 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 labeling 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. Once the models are trained, they are evaluated on unseen collected data to determine their effectiveness. In the following sections, more details are provided on the proposed federated learning architecture and how ML models are trained and evaluated using it.

1.3. Outline of the Thesis

The structure of this thesis is as follows. Initially, an examination of state-of-the-art research efforts is conducted, providing a comparison and identification of corresponding studies. This section also offers an overview of the background in this field, familiarizing the reader with foundational principles. The discussion then progresses to the tools and methodologies employed in the study. Subsequently, an explanation of the infrastructure design is provided. The next segment presents the results and practical applications of the research. Additionally, there is a detailed discussion section that explores every aspect of the specific area that this work focuses on, in comparison to other studies. Lastly, the work concludes by summarizing the main information that has been discovered through this research.

2. PRIOR RESEARCH

The use of self-learning systems, which employ unsupervised, semi-supervised, and supervised machine learning algorithms, is an effective method for dealing with attacks by processing both benign and malicious network traffic. Despite the many machine learning solutions offered in the literature, the applicability of these approaches to commercial network-based IDS (NIDS) is still in its early phases [11]. The primary issue with current machine learning solutions lies in their elevated false positive rate, coupled with substantial computational expenses. This problem primarily emerges because the classifiers locally learn the attributes of basic TCP/IP features.

Respectively, deep learning, a intricate subset of machine learning, acquires hierarchical feature representations and uncovers concealed sequential connections by utilizing TCP/IP data across multiple hidden layers. Deep learning has gained popularity in AI tasks for various applications like Natural language processing (NLP), speech recognition, image processing, etc [12]. In the field of cybersecurity, deep learning has found application in various domains, such as traffic analysis, classification of Android malware, network traffic prediction, categorization of encrypted text, intrusion detection, ransomware detection, malicious URL detection, malicious domain name detection, and anomaly detection, among others [13].

KDDCup 99 is a widely used benchmark dataset for academic research aimed at improving the success rate of intrusion detection [10]. The dataset was generated as the outcome derived from tcpdump data captured from the DARPA Intrusion Detection Evaluation Network in 1998 and was used for the third International Knowledge Discovery and Data Mining Tools Competition. The primary objective of the competition was to develop a predictive model with the ability to categorize network connections as either attack or normal. The attacks were R2L, U2R, DoS, and Probe.

For this competition, MADAMID served as the framework for constructing features [14]. This framework produces 41 features, 9 of which are basic features of a packet, 12 of them are content features, eight of them correspond to traffic characteristics, while the rest are related to host-based attributes. The entire dataset is provided, along with an additional 10% of complementary data. Decision tree and 1-nearest neighbor were two main algorithms for classifying the data that were used by contestants.

The majority of the published outcomes relied on 10% of this dataset, while a limited number of studies employed custom-built datasets. After the KDDCup 99 challenge, several published results have utilized feature engineering methods for dimensionality reduction. However, most recent studies have used the same dataset without custom-built features for machine learning classifiers, and their results are comparable to other contestants' outcome.

A comprehensive survey on ML intrusion detection with KDDCup 99 dataset was conducted by [15]. The investigation evaluated the efficacy of various machine learning algorithms in detecting intrusions by analyzing the KDDCup 99 dataset. According to the authors' findings, decision tree and Naive Bayes algorithms demonstrated strong performance on this particular dataset. The authors also noted that the dataset suffers from class imbalance, and methods such as undersampling and oversampling can be employed to tackle this problem.

The classification model presented in Agarwal's work [16] consists of two phases: P-rules, used to forecast the presence of the class, and N-rules, employed to predict its absence. It performed well on most categories in the KDDCup 99 dataset except for U2R. In [17], a relevance analysis was conducted for intrusion detection systems (IDS) using the KDDCup 99 dataset. The study measured feature relevance using information gain, identified and presented the most significant features associated with each class label.

A random forest approach utilized for detecting misuse, using anomaly detection and outlier detection techniques is presented by [18]. The research revealed that the misuse approach yielded superior outcomes compared to the results obtained from the KDDCup 99 challenge, and that the hybrid system enhanced the performance when merging anomaly detection and misuse [19, 20]. An intrusion detection algorithm using AdaBoost and decision stumps as weak classifiers was proposed by [21]. The approach exhibited superior performance compared to other methods, as it demonstrated a lower false alarm rate, higher detection rate, and faster computational speed. Nevertheless, it did not incorporate the incremental learning approach.

In [22], the Shared Nearest Neighbor (SNN) model was found to be the optimal algorithm exhibiting a high rate of intrusion detection. The study used a 10% dataset and showed that SNN outperformed K-means in detecting the U2R attack type. However, the research did not present any findings regarding the complete testing dataset.

Several studies have explored the utilization of Bayesian networks for intrusion detection. In one study [23], Naive Bayesian networks were employed, wherein the root node represented a class and the connecting nodes represented the connection features. Another study [24] investigated the application of Naive Bayes networks for intrusion detection by conducting a thorough analysis of experimental details. Their findings demonstrated that Bayesian networks exhibited comparable, and in some cases superior, performance in the probe and U2R categories when compared to the results of the KDDCup 99 challenge.

In addition to Bayesian networks, non-parametric density estimation methods have also been studied for intrusion detection. For instance, [25] studied the use of Parzen-window estimators, normal distribution, and Gaussian kernels. In comparison to studies utilizing an ensemble of decision trees, their findings yielded favorable results.

Moreover, a genetic algorithm was proposed by [26] that models spatial and temporal data used to detect intricate abnormal behavior. The algorithm was designed to streamline the process of identifying complex anomalous behavior.

Swarm intelligence methods have also been studied for intrusion detection. Study conduct by [27] investigated the application of ant colony clustering, ant colony optimization, and particle swarm optimization techniques in systems. These methods have shown potential for detecting anomalous behavior in intrusion detection systems.

2.1. Background

To fully comprehend the content of this work, we need to become familiar with the required background in this field. This section is dedicated to that purpose, so the

reader can go through all the necessary information to fully understand the dynamics and foundations.

2.1.1. Benchmark Datasets

In this portion, you can find information regarding the commonly utilized datasets by researchers and analysts to evaluate the effectiveness of their suggested approaches. As a result of concerns regarding security and privacy, the majority of datasets are not accessible to the public. Moreover, the datasets that are publicly available undergo extensive anonymization processes and may not accurately represent the diverse range of network traffic observed today. Consequently, there is still no definitive dataset that can be considered exemplary [28]. The details pertaining to different IDS datasets are elaborated upon in [29] and [30]. In [30], a comprehensive summary of the datasets available from 1998 to 2016 is presented.

We examine the advantages and disadvantages of the current datasets employed in NIDS (Network Intrusion Detection Systems) and HIDS (Host Intrusion Detection Systems). Generally, based on the contents of each dataset, they could be classified into the seven following groups:

1. Internet-connected devices
2. Electrical network
3. Virtual private network
4. Android app
5. Internet traffic
6. IoT traffic
7. Network traffic

In the following we will go through most of the datasets used in this field.

- The DARPA dataset, which utilizes audit logs and network traffic, was initially released in February 1998. The training dataset comprises seven weeks' worth of network-based attacks, whereas the testing dataset includes a two-week period of network-based attacks [4]. Nevertheless, according to Sharafaldin et al., this dataset fails to accurately depict real-world network traffic [31].
- With a training set comprising five million records and a testing set consisting of two million records, the KDD Cup '99 dataset is recognized as one of the most widely utilized and popular datasets for Intrusion Detection Systems (IDS). Each row of the dataset has forty-one distinct features (attributes) and is labeled as either attack or normal. The creation of this dataset involved the processing of tcpdump data extracted from the 1998 DARPA intrusion detection dataset. The Mining Audit Data for Automated Models for ID (MADAMID) was employed to derive features from tcpdump-captured data. MIT Lincoln Lab developed this

dataset by utilizing numerous UNIX machines and involving hundreds of users. The capturing of the network traffic continued for ten weeks. The dataset comes in two variations: the full dataset and the 10% dataset. The attacks in this dataset are categorized into four distinct types: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R) [32]. The features are organized into various categories as outlined below:

  - The basic features from UDP datagrams, TCP segments, and packet headers were extracted using pcap files obtained from tcpdump. Bro, which is a remodeled network analysis tool, was used to carry out this task. Features one to nine are related to this section.
  - The features described in this section are obtained by extracting information from the payload of TCP/IP packets contained within pcap files. In recent years, the analysis of features within the payload has emerged as a significant research focus. Instead of performing feature extraction, a deep learning approach was introduced to analyze the complete payload data [33]. The primary objective of these features is to detect attacks falling under the Remote to Local (R2L) and User to Root (U2R) categories. As an example, an essential feature in identifying malicious behavior within the payload is the presence of multiple failed login attempts. Sequential patterns are not evident in R2L and U2R attacks due to the nature of events occurring within a single connection. Features ten to twenty-two relate to this section.
  - Time-related features are obtained using a temporary window of two seconds. Connections are categorized as either "same host" or "same service" based on their characteristics observed within the past two seconds [10]. In occasions like slow probing attacks, a window of hundred connections is used on the same host to calculate the required characteristics. These features are referred to as either connection-based or host-based traffic characteristics. Features twenty-three to forty-one are related to this section.

- Kyoto dataset contains data from darknet sensors, web crawler, honeypots, email servers in addition to other network security measures. The records are from year 2006 to year 2015. There are a total of 24 features for each record, 14 of which come from KDD Cup'99 [34]. These 14 features hold significance as they originate from data extracted in Kyoto University.

- NSL-KDD is a modified version of the KDD Cup'99 dataset with redundant connection records removed which was introduced by Tavallaee et al. [35]. This dataset contains forty-one features, with attacks divided into four classes [36]. This dataset is normally used to avoid bias in machine learning algorithms. However, one limitation of the NSL-KDD dataset is that it does not accurately capture the characteristics of real-time network traffic profiles.

- Containing about two million records with forty-nine features, the creation of the UNSW-NB15 dataset was carried out by the Australian Center for Cyber

Security to overcome limitations observed in KDDCup99 and NSLKDD. The data is heterogeneous and The UNSW-NB15 dataset includes both normal network behaviors and various attack behaviors captured from live network traffic. The dataset was generated using IXIA Perfect Storm tool, which incorporates a collection of attacks, as well as the Common Vulnerability Exposures (CVE) repository. To generate both normal and malicious activities within the network, the IXIA traffic generator tool employed two servers. The task of capturing and storing network packets into pcap files of 1,000 MB was carried out by Tcpdump. The extraction of features was performed using both Bro (also known as Zeek) and Argus tools. C# was used for the in-depth analysis of each packet utilizing recently developed algorithms, the feature extraction process was conducted. The dataset encompasses the following categories of attacks [37]:

  - Worms
  - Port Scans
  - Shell code
  - DoS
  - Exploits
  - Fuzzers
  - Reconnaissance
  - Backdoor
  - Generic

  The data is available in two different formats as described below:

  - Complete two million network sessions
  - A percentage of the dataset that contains 82,332 records as train, and 175,341 records as test. Both train and test include 10 attacks. They also include 42 features [38].

- The CIC-IDS2017 dataset, established by the Canadian Institute for Cyber Security (CIC), comprises genuine network traffic as well as real-world attack instances [39]. The complete traffic data was gathered over a span of five days, commencing from July 3rd to July 7th, 2017. During this period, one day captured normal network activity, while the remaining days consisted of intentionally injected attacks into the network. Protocols such as SSH, HTTPS, FTP, HTTP from 25 users were used for normal traffic. In the analysis process using CICFLowMeter, details such as protocols used, IP addresses, timestamps and attacks were employed for the elicitation of the eighty network features. The dataset comprises the standard attack scenarios as described below [40]:

  - DoS
  - Brute Force for FTP and SSH

  - Botnet
  - DDoS
  - Web attack
  - Infiltration
  - HeartBleed

- The CSE-CIC-IDS2018 dataset comprises a variety of attack scenarios, including Heartbleed, Infiltration, DoS, Brute Force, DDoS, Botnet, Web attacks, and others [41]. The CICFlowMeter tool was utilized to analyze the network traffic and derive 80 network flow features.

- WSN-DS dataset was specifically designed for Wireless Sensor Networks and includes four categories of DoS attacks: Flooding, Blackhole, Scheduling, and Grayhole, which is comprised of 23 features in total.

- The DEFCON dataset is divided into two versions: DEFCON-10 and DEFCON-8. DEFCON-10 includes a range of non-probing and probing attacks such as bad packets, port sweeps, and port scans. DEFCON-8 contains only buffer overflow and ports scan.

- The CAIDA dataset comprises multiple datasets such as Internet traces 2016 from CAIDA, DDoS from CAIDA, and RSDoS attack metadata which was provided by the Center of Applied Internet Data Analysis. The RSDoS attack metadata contains randomly spoofed DoS attacks deduced from the backscatter packets gathered by the UCSD Network Telescope.

- The CDX dataset established during a network battle competition, encompasses both regular and malicious TCP communications on vulnerable network services, including those susceptible to buffer overflow attacks [42]. Furthermore, the dataset includes four types of vulnerable servers from CDX 2009: OpenFire Chat FreeBSD, Apache Web Server Fedora 10, Postfix Email FreeBSD and BIND DNS FreeBSD.

- TWENTE has a subdivision employing three IP protocols: ICMP, UDP, TCP. It contains over fourteen million flows and over seven million alerts.

- Being proposed by Jazi et al. [43], CIC DoS dataset includes application layer DoS attacks. We have low and high volume attacks for this dataset. Examples of high-volume HTTP attacks comprise:

  - Goldeneye
  - ddossim
  - hulk

  On the other hand, instances of low-volume HTTP attacks encompass::

  - read

  - send headers
  - send body
  - RUDY
  - Slowloris

- The ISCX dataset consists of a compilation of seven days' worth of network activity, encompassing the following:

  - Benign
  - DoS
  - Brute Force
  - DDoS
  - Port Scanning
  - Botnet
  - Web
  - Infiltration
  - SQL Injection
  - R2L
  - U2R

  Shiravi et al. [44] created this dataset, which includes two classes of profiles. The first class is designed to describe attack scenarios, while the second class encapsulates the extracted mathematical distributions of specific entities.

- The ADFA2013 dataset, introduced by Creech and Hu [45, 46], was developed for attacking the Ubuntu OS using vectors and payloads. The dataset's structure contains 4373 normal data points, 833 validation data points, with 10 attack data. The payloads include C100 Webshell, Java-based Meterpreter, password brute force, and Linux Meterpreter.

- Moustafa [47] has presented a heterogeneous dataset known as "TON_IoT," which comprises real-world IoT network activities from cloud, edge, and fog layers. This dataset is designed for evaluating the credibility of new systems and consists of sophisticated scenarios. The complete dataset was generated at the IoT lab of the University of New South Wales, utilizing the NSX vCloud NFV platform. It includes both raw and processed data collected from Telemetry datasets of IoT services, Operating Systems datasets, and network traffic. The dataset encompasses nine specific types of attacks as the following:

  - Scanning
  - Denial of Service
  - Distributed Denial of service
  - Ransomware

- Backdoor
- Injection
- XSS
- Password Cracking
- MITM

2.1.2. Datasets' Limitations

Privacy and security issues hinder the publication of datasets that represent current network traffic attacks, while publicly available datasets are often extremely anonymized, rendering them less pragmatic. Despite criticisms of the KDDCup 99 public dataset, it has been used for many research studies on NIDS over the years. Through a comprehensive examination of the contents, [35] identified simulated artifacts and discrepancies within the simulated network traffic data. They claimed that various network attributes, such as TTL, TCP window size and options, and remote client address, have limited range and are small in the KDDCup 99 datasets but exhibit growing range and are larger in real-world network traffic environments.

In KDDCup99, machine learning models face limitations in accurately detecting R2L and U2R attack types [48]. None of the classifiers were capable of improving the attack detection rate using this dataset. It is argued that attaining a high detection rate often involves the creation of a new dataset by amalgamating both testing and training datasets. Moreover, machine learning classifiers perform poorly with this data as many of the "snmpgetattack" belong to the R2L type [49].

As mentioned earlier, KDDCup 99 is a widely used, reliable benchmark dataset for security-related tasks. NSL-KDD was proposed by [35] to solve the inherent problems pertaining to KDDCup 99. Redundant connection records were eliminated from both train and test. This was specifically done to protect the classifier against bias towards connection records with more frequency. However, because NSL-KDD fails to accurately represent network traffic data observed in real-world scenarios, it falls short in addressing the issues presented by [50]. Extra features were added to enhance its performance in detecting attacks.

Even though automated network traffic using honeypots was used in the Kyoto dataset, its normal traffic was not obtained from network traffic data in real-world scenarios. Furthermore, it lacks the inclusion of false positives, which aid in minimizing the number of alerts for the network administrator.

Using two profile systems, one for normal activities and the other for generating attacks, [44] produced a new dataset. However, most of the attacks lack the features of real-world statistics, and the traffic does not contain HTTPS protocol traffic. Incorporating the concept of profiles that encompass comprehensive information regarding protocols, applications, intrusions, and network details, [50] proposed UNSW-NB15. On the other hand, in an effort to offer a standardized dataset that can serve as a reference for the research community, [39] provided a dataset that contains both malicious and benign network activities. They also validated the network traffic features in experiments toward detecting various attacks.

3. METHODS, TOOLING AND DEVICES

The details of the methods, tools, and devices that were utilized for this work will be illustrated in this section.

3.1. Network Monitoring

Intrusion Detection Systems (IDS) are designed to identify various types of attacks, such as malware infections, network scanning attempts, port scanning, and other suspicious activities. When an Intrusion Detection System (IDS) identifies a potential threat, it can either notify security personnel or initiate automated actions to prevent or mitigate the attack. [10]. Two primary categories of IDS exist: network-based IDS (NIDS) and host-based IDS (HIDS) [9], which are other two important terms in our jargon. NIDS monitors network traffic, while HIDS monitors activity on individual systems. IDS plays a crucial role as a component of network security and is frequently utilized alongside complementary security measures, such as firewalls and antivirus software, to enhance overall network security. [10].

Suricata is open-source network intrusion detection and prevention system (IDS/IPS) that is freely available. This system can actively monitor real-time network traffic and promptly notify administrators of any suspicious activities or potential security threats. Initially launched in 2010, it is written in the C programming language [51]. Suricata is designed with a strong emphasis on scalability, enabling it to inspect high-speed network traffic on multiple network interfaces simultaneously, and it provides extensive protocol support, encompassing a broad array of protocols such as HTTP, SMTP, DNS, TLS/SSL, and others. In addition to rule-based and signature-based detection techniques, it possesses the capability to conduct advanced threat detection and analysis through behavioral analysis and anomaly detection methodologies. Suricata is widely used in various industries, including government, financial services, healthcare, and telecommunications, to strengthen network security and safeguard against cyber attacks. It is also compatible with many other security tools and can be integrated into existing security infrastructures.

3.2. The PanOULU Public Network

As the goal of the thesis is to develop a Distributed Intrusion Detection System (IDS) for public networks, it is essential to possess a comprehensive understanding of the network itself and its extensive coverage area. PanOULU is a wireless network infrastructure that provides Wi-Fi connectivity in Oulu, Finland. It is a network publicly available for anyone to use freely, and it covers a large part of the city center as well as some surrounding areas.

The network is operated by the City of Oulu in collaboration with the local university, research institutes, and businesses. It was originally launched in 2006 and has since been expanded and upgraded to provide faster and more reliable connectivity.

PanOULU uses a mesh network architecture, which means that the Wi-Fi access points are interconnected wirelessly to establish a network. This allows users to move

around the city and maintain a stable connection to the network without having to constantly reconnect to different access points.

The network is designed to provide both indoor and outdoor coverage and is intended to support various applications and services, such as public transportation, tourism, and business activities. PanOULU is also used as a research platform for developing and testing new wireless technologies and applications.

Juha Tiensyrjä et al. develop a novel multiplayer game with pervasive location awareness utilizing the panOULU [52]. The game requires players to conquer real-world access points (APs) of the panOULU network to score points, and implementing the game as a web service is facilitated by the network's built-in support for session mobility, which serves as a key functionality. Examining the impact of the game on players' network usage, the study focuses on session mobility and its relevance to network design.

The panOULU network was the basis for the development of the Luotsi web mash-up application by Hannu Kukka et al. [53]. The objective of the application is to deliver users with timely and pertinent information that aligns with their present geographical position to improve their experience by enabling easy access to information regarding neighbour service and sites etc. Specifically, Luotsi leverages the integration of diverse information feeds and real-time user positioning to enhance its functionality, provided by panOULU, to provide valuable insights about the user's present circumstances.

Timo Ojala et al. aim to highlight the relevance of the panOULU by presenting analysis on its ability to freely provide Internet access to the general public [54]. The paper discusses the founding and evolution of panOULU, its usage statistics, and the motivations behind its establishment, with specific attention to the involvement of the City of Oulu. Furthermore, panOULU is used as an example to illustrate the application of the Triple Helix Model (THM) in innovative studies and as a model for tackling typical difficulties encountered by urban wireless networks. Ultimately, panOULU serves as an embodiment of successful collaboration between different sectors in pursuit of a common goal.

3.3. Operating Systems in the Network

To guarantee the efficiency of the framework that utilizes the panOULU network, it was crucial to obtain a comprehensive understanding of the operating systems used by devices within the network. To achieve this, the p0f v3 tool was employed to passively collect information about the operating systems of devices that were in the same subnet as the IDS.

As a passive tool, p0f v3 examines the network traffic patterns of remote hosts to identify the operating system being used. This includes details such as the user's language, hardware type, and the version of the operating system and time zone. Network administrators can use this information to better understand the devices on their network, identify any potential security risks, and take appropriate security measures.

The use of p0f v3 is highly beneficial in ensuring that the appropriate security measures are in place for devices on the network. Administrators can also take proactive measures to prevent attacks that target specific operating systems.

The following figures depict operating systems both within and outside of the panOULU network, including those from servers on the internet. The presented operating systems are determined by the distinctive combination of source and destination IP addresses, ensuring that no OS is counted multiple times for a single device that connects to multiple other devices. However, this approach is not a perfect indicator of uniqueness, as one device could connect to multiple machines and servers with different IP addresses. As a result, it is possible that some of the reported OSes may be counted multiple times, but the figures still provide a rough estimate of the frequency of OSes in the network.

Figure 1 presents a bar plot of the OSes in the network, while Figure 2 provides a detailed breakdown of the number of each OS. These figures provide insight into the OS types in the network and its connected devices and servers.

3.4. Required Tools

Five ESXi hosts were used to install the required virtual machines. Different devices and machines with varying computational capacity and disk space were used to install ESXi hosts. Furthermore, the vSphere client appliance was installed on one of the machines to enable connection to the server, manage installations and handle inventory objects from a central location using vCenter Server Management Software. The following section provides a detailed description of the necessary steps that were taken to set up the entire infrastructure.

3.5. ESXi Host Installation

To install an ESXi host on a device, the installation link [55] is visited. The image is then burned onto a USB stick using the Rufus application. After the setup process is completed, a static or dynamic IP address can be assigned. In this case, DHCP assigned IP addresses were used. The ESXi host can then be accessed using the assigned IP address from anywhere within the panOULU network.

3.6. Installing Virtual Machines on ESXi Hosts

After entering the IP address of the desired ESXi host and providing our own setup username and password, a dashboard is presented to manage and control the machine. A glimpse of the dashboard is provided in Figure 3.

Accurate time synchronization is crucial for effective management of ESXi hosts through vCenter Server. Vital to this is ensuring that all ESXi hosts are using the same time server. The NTP settings can be edited by navigating to the "System" tab under the Host option in the manage section of the navigator pane. The time server address, such as `time.google.com`, can be entered here.

To upload the image of the operating system, navigation to the "Datastore browser" button in the "Datastores" tab of the storage section in the navigator pane is required. The "Upload" button can be used to upload the image of the operating system.

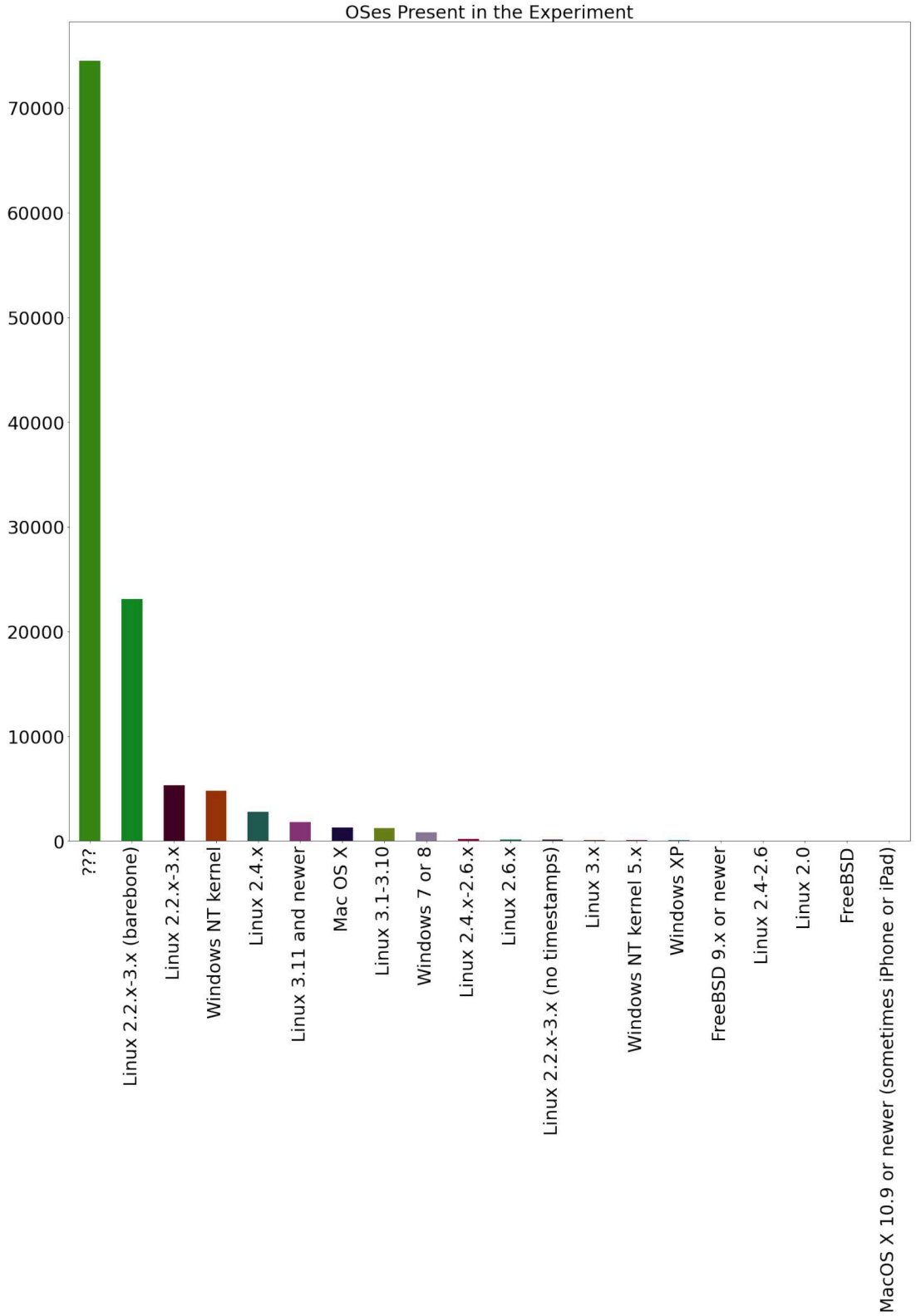

Figure 1. The frequency of the OSes involved in the experiment.

	os
???	74492
Linux 2.2.x-3.x (barebone)	23106
Linux 2.2.x-3.x	5340
Windows NT kernel	4780
Linux 2.4.x	2805
Linux 3.11 and newer	1814
Mac OS X	1273
Linux 3.1-3.10	1213
Windows 7 or 8	803
Linux 2.4.x-2.6.x	214
Linux 2.6.x	131
Linux 2.2.x-3.x (no timestamps)	116
Linux 3.x	81
Windows NT kernel 5.x	59
Windows XP	55
FreeBSD 9.x or newer	3
Linux 2.0	2
Linux 2.4-2.6	2
MacOS X 10.9 or newer (sometimes iPhone or iPad)	1
FreeBSD	1

Figure 2. The frequency of the OSes involved in the experiment.

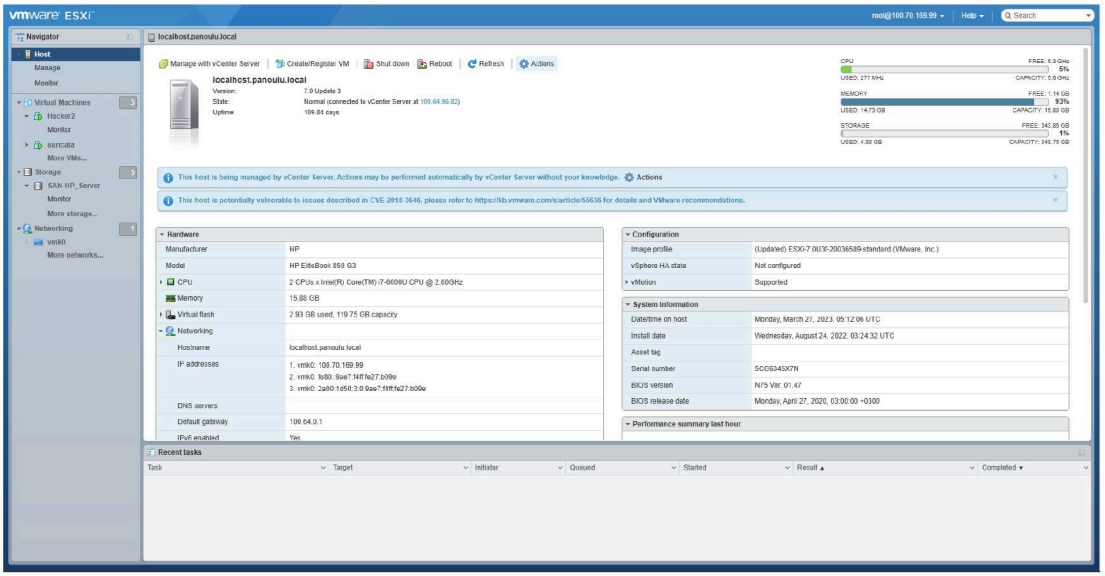

Figure 3. ESXi host's dashboard.

Following the successful upload of the operating system's image, the next step involves its installation on a virtual machine. Creating a virtual machine necessitates navigating to the "Virtual Machines" section of the navigator and selecting the "Create/Register VM" button. The wizard menu is then utilized to customize the virtual machine in preparation for the installation of the previously uploaded operating system's image. Upon completing these steps, the virtual machine is capable of running, and the operating system's installation can start.

3.7. Installing VCenter Server Appliance

Managing VMs running on ESXi host devices can be challenging. To simplify this process, vCenter Server Management Software is necessary. The vCenter Server Appliance image must be downloaded and mounted on a separate machine. The setup process is initiated, and the installation path, "vcsa-ui-installer."

The appropriate folder is selected for installation, depending on the operating system. In the case of Windows operating system, the win32 folder is chosen. Once the installation menu is executed, the process is straightforward. The required fields, such as the IP or FQDN of the ESXi host and its credentials, are provided for the vCenter server appliance to be installed on the designated ESXi host.

Accessing the installed VMware vCenter Server is possible through its IP address or FQDN. If the IP address is assigned using DHCP, the IP address can be determined by navigating initially to the IP address of the ESXi host that is hosting the installed vCenter Server. Afterwards, the IP address assigned by DHCP can be viewed by clicking on the VMware vCenter Server VM in the "Virtual Machines" section of the navigator pane. In this case, the appliance access utilized the DHCP-assigned IP address.

The DHCP-assigned IP address was used for appliance access, and the IP was determined by clicking on the VMware vCenter Server VM in the "Virtual Machines"

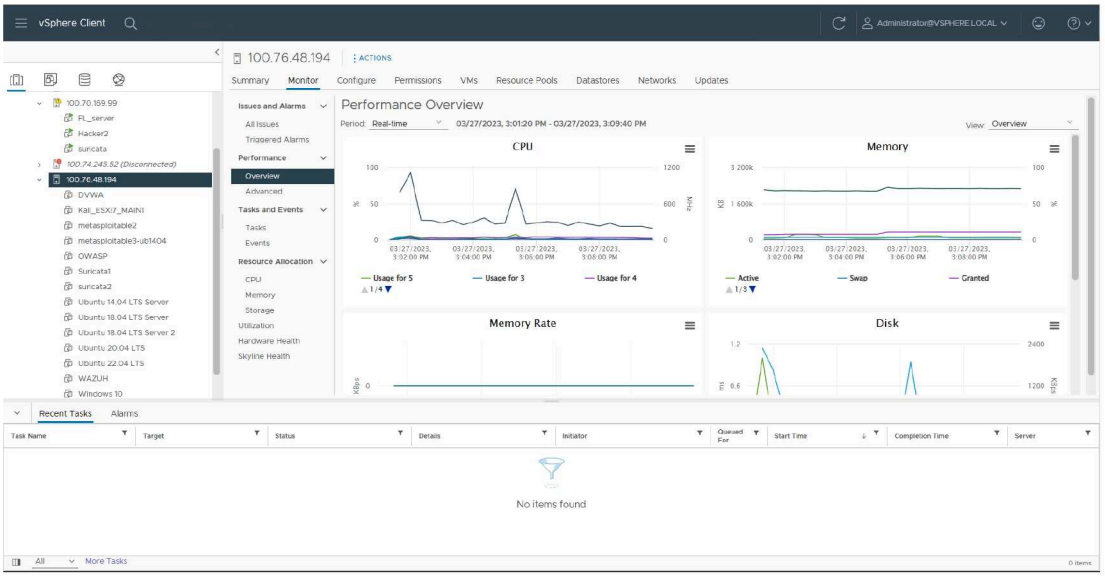

Figure 4. Glimpse of vCenter's dashboard.

section of the navigator pane after navigating to the IP address of the ESXi host responsible for hosting the installed vCenter Server. Figure4 provides us with an overview of the dashboard of vCenter.

3.8. Adding ESXi Hosts to VCenter Server

After installing the vCenter Server appliance, it is crucial to incorporate the ESXi hosts into the appliance to enable unified management from a centralized location. To add an ESXi host to the vCenter Server appliance, a data center needs to be created first. The data center can be created by right-clicking on the FQDN name or IP address of the appliance.

Once the data center is created, the "Add Host..." icon should be selected by right-clicking on it. The wizard menu will then be presented, and after filling out the required fields, the ESXi host will be added to the data center section of the vCenter Server appliance. With this functionality, all the resources of the ESXi hosts, including their network configuration and data stores, can be accessed from a central location.

By following the aforementioned steps, one can seamlessly manage multiple ESXi hosts through the vCenter Server appliance. Instead of individually managing each ESXi host manually, the vCenter Server appliance can be used to centrally manage all hosts, resulting in both time-saving and simplification of the management process.

3.9. Adding SAN to the VCenter Server

The proposed architecture for the panOULU network requires significant storage capacity to orchestrate all the virtual machines (VMs). To address this need, it was decided that a Storage Area Network (SAN) would be integrated into the network. This will allow for the hosting of VM storage using Logical Unit Numbers (LUN),

which can be accessed from any part of the network. To configure the storage, the Synology DiskStation needs to be set up properly. Once this is done, the ESXi hosts can incorporate the necessary data storage.

To create a LUN, a volume needs to be created in the disk station. A volume can be created by accessing the storage manager application at the IP address of the disk station. A storage pool is then created to enable volume creation. Once the desired volume has been created, the SAN manager application can be opened. In the LUN section of this application, a desired LUN is created. This LUN will be utilized by ESXi hosts as a data store.

To add a data store using LUN, the ESXi host's dashboard needs to be visited. The Adapters tab in the storage section should be clicked on, followed by the Software iSCSi button. The IP address and port number of the disk station should be added in the dynamic area, and a vmkernel adapter should be added for managing the network communications in the port binding area. Once these settings have been configured, the datastores tab can be clicked on. From there, a new data store can be selected, and the "New VMFS data store" option can be chosen. On the following page, the created LUN can be seen and added to the data stores. The desired VMs can then be stored in this data store.

3.10. Creating Distributed Switch

To enable monitoring of traffic from all virtual machines hosted on various ESXi hosts, it is necessary to create a distributed switch within the vCenter Server. To do so, one can click on the network icon in the inventory section of the vCenter Server. Next, right-click on the created data center and select the "Distributed Switch > New Distributed Switch" option. The required information should be filled out in order to create the distributed switch.

Once a distributed switch has been created, available Network Interface Cards (NICs) from various ESXi hosts need to be added to it. To do so, the created distributed switch should be right-clicked and the "Add and Manage Hosts..." option selected. In the wizard menu, ESXi hosts with available NICs should be chosen and free NICs assigned to the desired uplink of the distributed switch.

After all these steps have been completed, the distributed switch will be deployed on the selected ESXi hosts. It is also possible, at one of these steps, to migrate the switch of existing virtual machines to this distributed switch.

3.11. Adding VMs' Network Card to Distributed Switch

When creating a new virtual machine, the deployed distributed switch in the ESXi host can be selected instead of the internal networking in each ESXi host. This can be accomplished via vCenter Server by right-clicking on the desired ESXi host and choosing "New Virtual Machine". In the "Customize hardware" section of the wizard menu, the port group of the distributed switch to be used for the network hardware can be selected. With this configuration, the network settings of the VMs will be managed via the selected distributed switch.

3.12. Port Mirroring

To listen to the traffic of all VMs in one or multiple locations, port mirroring can be utilized. The process involves clicking on the network icon in the inventory section of the vCenter Server appliance, followed by clicking on the created distributed switch and selecting the configure section. In the configure section, the "Port Mirroring" option in the settings pane can be chosen, followed by clicking on "NEW..." A wizard menu will appear, where the "Encapsulated Remote Mirroring (L3) Source" option should be selected. Then, the desired VMs that need to be listened to can be chosen in the next pages. In the end, the IP address of the machine that will listen to all the traffic of the selected VMs should be provided.

3.13. Various Levels of Compromise

Attackers are typically defined as unauthorized users who attempt to initiate intrusions. Malicious activities can be performed by an attacker using a remotely accessible computer. To accurately detect an intrusion, it is necessary to understand the methods used by an attacker.

An attack generally consists of five phases: Reconnaissance, Exploitation, Reinforcement, Consolidation, and Pillage [10]. It is possible to detect an attack during the first three stages. However, once the attacker progresses beyond the third stage, the system is fully compromised. Distinguishing between benign and malicious behavior becomes extremely challenging at that point. To gain a better understanding, each of these stages will be explored in more detail.

In the reconnaissance step, information is gathered by an attacker to collect data on reachable devices and hosts, their running services, OS versions, and running applications. In the exploitation step, a specific service is exploited by the attacker to hijack the targeted device. Breaching, abusing (like dictionary attacks or stolen passwords), or subverting (like SQL injection) or some examples for this.

After gaining access to the targeted system, a camouflage phase is initiated where necessary tools and services are installed by the attacker to exploit the privileges gained in the reinforcement step to use the applications that are available via the hacked account. In the consolidation step, the accessed system is fully controlled by the attacker via the backdoor that is employed for communication in this step. In the final stage, the pillage step, the attacker steals data and computational resources or could even impersonate.

It is almost always the case that there are some bugs in either hardware or software. These bugs, induced by human faults, could lead to security vulnerabilities. Data Integrity, Confidentiality, and Availability are the three essential components that form the pillars of information security. Additionally, we can include accountability and authenticity. Attacks on confidentiality usually relate to passive attacks, such as scanning and availability of addresses. Exploiting available addresses could make network resources down, by attacks like Denial of Service (DoS) or Distributed Denial of Service (DDoS), which can prevent normal users from accessing them. Eavesdropping is often challenging for IDS systems, and locally within a system or over a network, probing can be performed.

With this background, a set of actions that may compromise the confidentiality, availability, data integrity, or other security policies of a targeted resource or system can be defined as an attack.

3.13.1. Adversary Emulation

The proposed architecture utilizes the panOULU network by deploying the distributed switch and connecting edge, fog, and cloud tiers. Apart from the benign packets that are naturally flowing in the network, this work also requires malicious packets. This is to make sure that the dataset contains both benign and malicious packets that could be utilized by the machine learning models.

The reconnaissance phase was implemented to generate the malicious packets. This was done by assigning two VMs in the fog layer to act as attackers. Nmap was used to scan the devices in the fog layer and edge layer, which were set up in the cyber range. Used by network administrators and security experts, Nmap is a popular tool for network exploration and security auditing. Creating a map of the network, it is utilized to discover hosts and services on a computer network. Identifying open ports, running services, and the operating systems used by hosts on a network are among its capabilities. The scanning process for this work included ping scanning using ICMP packets and host scanning by skipping port scanning during host recovery. After each step of scanning the specified devices and systems, a 5-minute break is given, and then again the process loops indefinitely.

It should be emphasized that the scan is confined to devices within the cyber range, ensuring a limited scope for the analysis, and it does not extend beyond this boundary. This is important to ensure compliance with GDPR and other regulations.

3.14. ML Models Employed on This Work

There are four ML models that are employed in this work, and later in the section, certain metrics will be introduced to assess their performance. The following enumerates these employed models along with their terse description:

1. Logistic regression is a statistical method used for binary classification. The probability of the binary outcome is estimated by the model through the application of a logistic function to a linear combination of the input variables. It also can be used for both numerical and categorical input variables.

2. Perceptron is an artificial neural network model used for binary classification. It takes a set of input values, applies weights to each input to produce a binary output. The algorithm is trained using an iterative process to adjust the weights until the classification error is minimized. It is a linear classifier and works well when the data is linearly separable, but may not perform well on complex datasets

3. Another binary classification model used for binary classification is Passive Aggressive Classifier. It is a variant of perceptron and is designed to update

its model parameters in an aggressive manner when it misclassifies an example, while remaining passive when it makes a correct classification. The algorithm is known for its fast training speed and ability to handle large-scale datasets. It is commonly used in online learning scenarios where the model is updated in real-time.

4. Stochastic Gradient Descent (SGD) Classifier is another binary classification model. It is a variant of gradient descent optimization algorithm that in which the model parameters are adjusted by making incremental changes in the direction opposite to the gradient of the loss function, using randomly selected examples from the training set. The algorithm is particularly well-suited for large datasets, as it updates the model incrementally and does not require the entire dataset to be loaded into memory. Stochastic Gradient Descent (SGD) finds extensive usage in various fields, including deep learning and other machine learning applications, where it is employed for the purpose of training a wide range of models, including logistic regression, support vector machines, and neural networks.

3.15. Evaluation Metrics

Ground truth is required to evaluate the estimates of various evaluation metrics. The ground truth is comprised of a mixture of connection records that are categorized accordingly as either malicious or benign for binary classification. The following terms are employed to evaluate the quality of the classification models:

- False Negative (FN) refers to a situation where a true positive outcome is mistakenly classified as negative.
- True Negative (TN) refers to a situation where a true negative outcome is correctly classified as negative.
- True Positive (TP) refers to a situation where a true positive outcome is correctly classified as positive.
- False Positive (FP) refers to a situation where a true negative outcome is mistakenly classified as positive.

We also have the following commonly used evaluation metrics:

- **Precision** refers to the measure of the accuracy of positive predictions. Higher precision rate indicates better machine learning model.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

Precision $\in$ [0; 1]

- **Recall** refers to the measure of the ability to identify positive instances correctly. A higher recall indicates a superior machine learning model.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Recall $\in$ [0; 1]

- **Accuracy** refers to the measure of correct predictions over the total number of predictions made. It serves as a reliable metric for evaluating the performance of a machine learning model on a test dataset with balanced classes. Higher accuracy means better machine learning model.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Accuracy $\in$ [0; 1]

- **F1-Score** or **F1-Measure** combines precision and recall into a single metric to assess the model's overall performance.

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

F1-score $\in$ [0; 1]. The higher value means better machine learning model.

- **Receiver Operating Characteristics (ROC) curve** is a graphical representation of the trade-off between true positive rate and false positive rate. The evaluation metric used to compare machine learning models is the Area Under the ROC Curve (AUC), which represents the extent of the area beneath the ROC curve.

$$AUC = \int_0^1 \frac{TP}{FN + TP} \, d\frac{FP}{FP + TN} \tag{5}$$

Higher value is an indication for better machine learning model.

4. INFRASTRUCTURE DESIGN

This section presents the proposed architecture intended to generate a novel dataset that accurately mirrors the modern edge networks and IoT devices employed by smart cities and organizations in today's context. The details of the federated learning developed and implemented on top of this architecture will also be explored.

The proposed architecture is based on IoT systems and interacting networks with layers of fog and edge to represent the realistic implementation of today's networks. The design of this architecture is illustrated in Figure 5. Furthermore, the cloud layer in the architecture is provided by the CSC — the IT Center for Science, owned by the Finnish state and higher education institutions, is a Finnish center of expertise in information technology.

Fog computing and edge computing provide similar on-premise services, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). They offer end-users and organizations the benefits of managing their data or IoT systems and empowering intelligence and analytics that are close to them in lieu of transferring a vast quantity of data to the cloud sources.

This architecture aims to generate a dataset that accurately reflects the current state of smart cities and organizations in the world. Federated learning is implemented on top of this architecture to enable collaborative machine learning between devices in the 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.

There is a difference between fog and edge computing based on the location of computing, intelligence, and analytics components. The edge layer places these components in physically sensible devices, such as IoT devices, laptops, personal computers, and physical servers. In contrast, analytics, computing and intelligence for the fog layer is considered in Local Area Networks (LANs) where data is routed through gateways to their intended destinations.

In the proposed architecture, the edge layer encompasses IoT devices, servers, routers, laptops, Raspberry Pi etc. where the fog is intended for intelligence, computing, and analytics using Hypervisor technology and LAN. The cloud layer, in this case, is used for hosting the Flower server — an open-source framework for federated learning systems. Flower enables data to be stored on distributed devices or servers, and the model is trained decentrally [56]. Flower clients access the server in the cloud.

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

- The fog layer is a virtualized space that controls services and virtual machines using the VMware vSphere platform. This platform includes frameworks for Network Function Virtualization (NVF), Software Defined Networking (SDN), and Service Orchestration (SO) in 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 fresheners, etc. and host systems such as servers,

workstations, and laptops used for connecting gateways and IoT devices to the panOULU network, and from this network to the internet.

VMware vSphere hypervisor technology is deployed on multiple hosts at the edge layer for managing VMs operating on the fog.

- 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. The learning weights of these individual ML models are sent to the Flower server in the cloud to be aggregated and averaged. Then, the end result from the Flower server in the cloud is sent back to the Flower clients in the fog layer. This is a continuous and automated process without any 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 intended functionality of the proposed architecture relies on the significant contribution of each of these technologies. To provide a better understanding of how these components work together, let us take a closer look at the functions of each layer and the involved technologies.

1. Service Orchestration (SO) refers to the coordination and management of multiple services to achieve a specific business process or goal to establish and deliver end-to-end services [47]. In the proposed architecture, vCenter server appliance was deployed on one of the hosting ESXi 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 for SO, as well as the monitoring of running systems and dynamic workload optimization along with proactive issue avoidance.

2. SDN enables the development of virtual networks that possess the functionalities of physical networks [57]. To improve network performance with dynamic and programmable network configurations, SDN is utilized. 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 machines that operate concurrently to provide various services.

3. By decoupling network functions from dedicated hardware, NFV enables virtualization and dynamic allocation of network services [58]. In order 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 the transmission of data for both benign and malicious activities by enabling communication between the different layers.

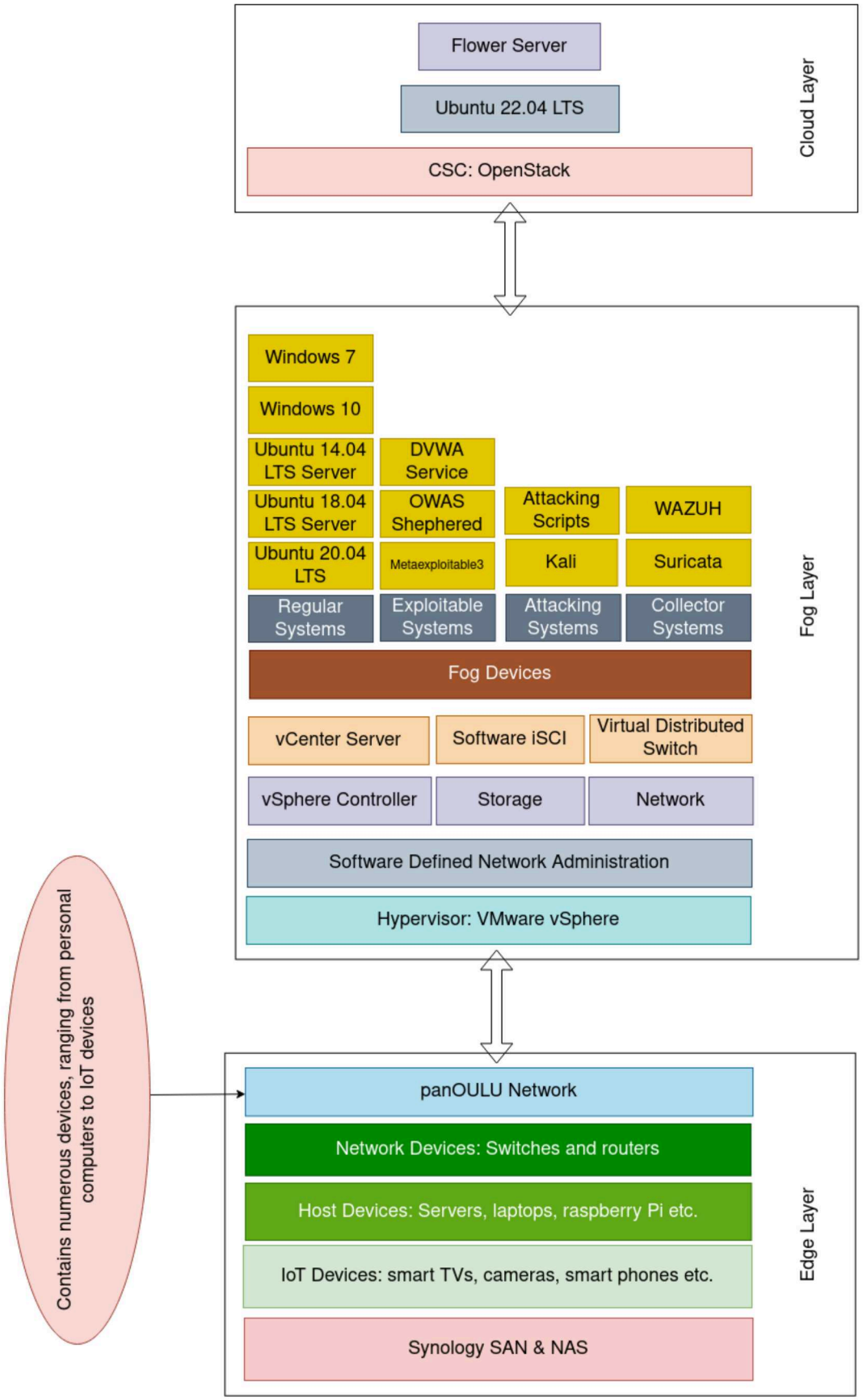

Figure 5. A novel and dynamic architecture is proposed for assessing security applications in edge networks and the Internet of Things (IoT).

Virtualizations of the proposed architecture are included in the fog. NFV, fog devices, and SDN are deployed and managed by the VMware vSphere, and the detail is as follows:

- SDN management and hypervisor technology were configured and deployed using the network and controller platforms.

    1. 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 for the management of vSphere systems, ensuring security through the inclusion of vSphere replication and data protection features. In other words, it serves as our service orchestration (SO) tool.
    2. 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).
    3. 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 5).

- 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, which are referred to as clusters.

- 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. The functionalities of these components, as shown in Figure 7, are illustrated below.

    - Regular Systems: are a set of systems that operate normally 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 get their IPs through a DHCP server automatically. The regular systems have the following operating systems installed on them:

        * Windows 7
        * Windows 10
        * Ubuntu 18.04
        * Ubuntu 20.04
        * Ubuntu 14.04

- Exploitable Systems: These are systems that are intentionally vulnerable so that they can be easily hacked. They resemble real-world systems that are never updated and have real security vulnerabilities.

  It is noteworthy to know that these naive systems were cordoned off by a firewall so that they could not have outbound traffic to the rest of the network. This is for security concerns so that no real attacker could exploit other systems in the network via these exploitable systems. 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.

- Attacking Systems: These are the systems specifically used for hacking in the testbed architecture. All devices shown in the proposed cyber range architecture can be considered as possible targets for attacking scenarios. In this context, a probing attack (scanning) is launched only for the devices in the fog layer and edge layer of the cyber range. It is important to limit other attack scenarios so that they do not affect all the devices in the network.

- Collector Systems are responsible for capturing and collecting network traffic from the entire testbed architecture. Suricata and WAZUH were deployed for this purpose. Suricata captures network traffic in pcap format that is mirrored to it from both the SPAN port of the switch and the port forwarding capability of the virtual distributed switch in the vCenter. It then sends the captured logs to WAZUH for further visualization and indexing in the ELK stack.

  Furthermore, for packet-level analysis, the Tshark tool is employed to extract valuable network packet features, which are subsequently utilized

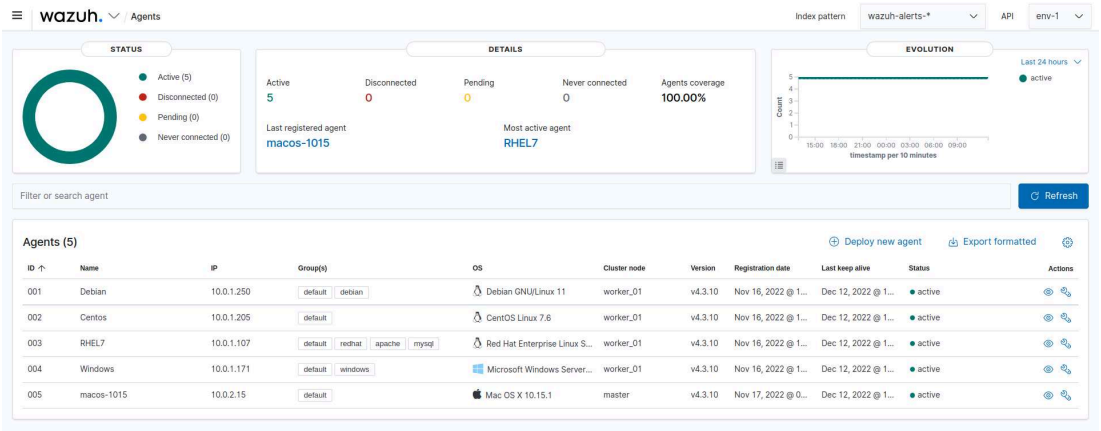

Figure 6. WAZUH's dashboard [59].

in conjunction with machine learning and deep learning algorithms. The identified patterns and deviations in network traffic, derived from the extracted features, can be leveraged to detect potential malicious activity within the network.

– Federated Learning Systems: These are the systems that employ the whole proposed architecture and infrastructure to provide the end result of recognizing malicious patterns done 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 private 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.

  Based on the attackers' IP addresses and the timestamp of the attacks, these packets are labeled as either benign or attack type (probe packets). After labeling, 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 process can continue for as long as desired, as it is configurable during the initialization phase of the whole federated machine learning process.

Figure 8 illustrates the operational mechanism of federated learning within the presented architecture. Each suricata machine in the fog layer hosts an individual

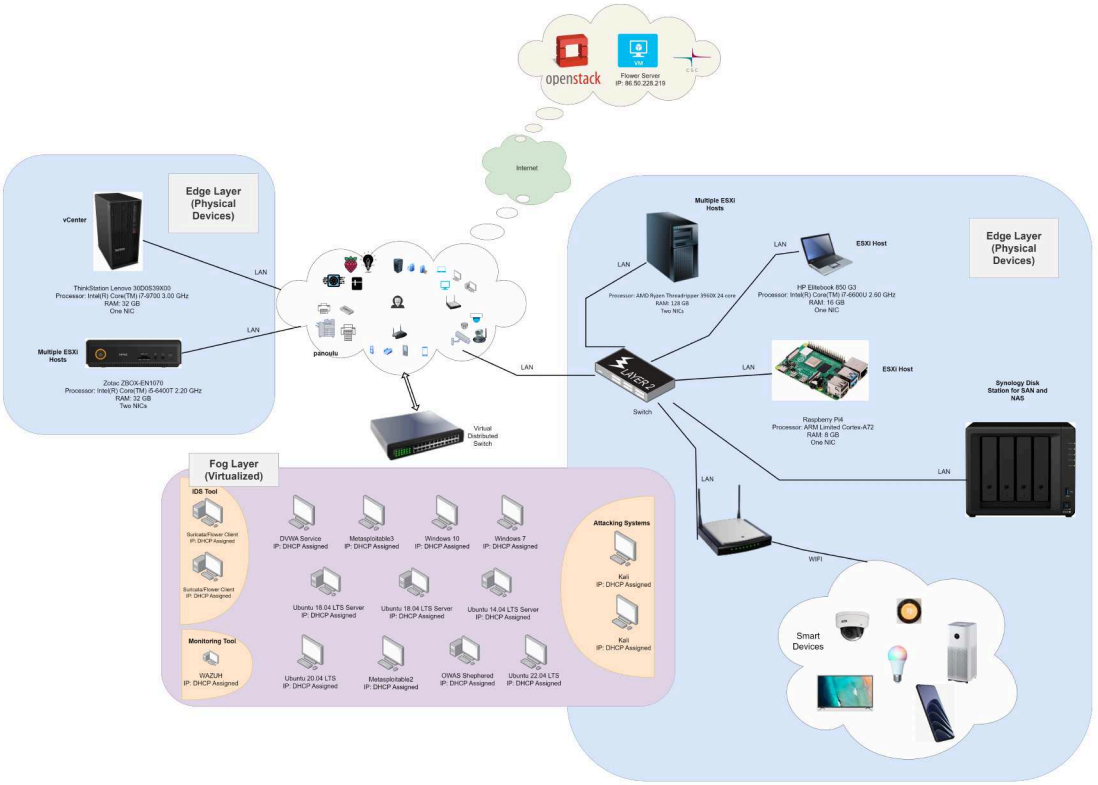

Figure 7. Cyber range for the test bed

FEDerated Machine Learning (FED-ML) client. The FED-ML clients collect streaming data by storing network packets as pcap files, leveraging the panOULU public network. It is worth noting that no human intervention is required in the process.

After preprocessing and labeling the network data, it is fed into the FED-ML models to be trained locally. In each iteration of the process, the learned model's weights are transmitted to the Flower server located in the cloud by each individual FED-ML client. The blue and green dotted and dashed lines in Figure 8 illustrate this transmission and the path taken in the architecture.

The Flower server in the cloud layer collects all the weights from the connected FED-ML clients and averages them. The updated weights are then sent back to the clients, as shown by the dashed red line in Figure 8. This whole process could continue as long as desired.

In this architecture, the federated learning process is an ongoing one, as the panOULU public network is being leveraged. The FED-ML clients are guests of the suricata machines, which not only function as IDS/IPS machines but also host the FED-ML clients. The clients constantly read the streaming pcap files provided by suricata and extract useful features from them. Based on the attackers' IP addresses and the time stamp of the attacks, the packets are labeled either as benign or as attack type (probe packets).

The IP addresses are stripped from the data to represent the real-world situation for the ML model, so that it does not learn patterns biased by IP addresses. 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 CSC was chosen as the cloud computing platform to deploy the Flower server. This enables the

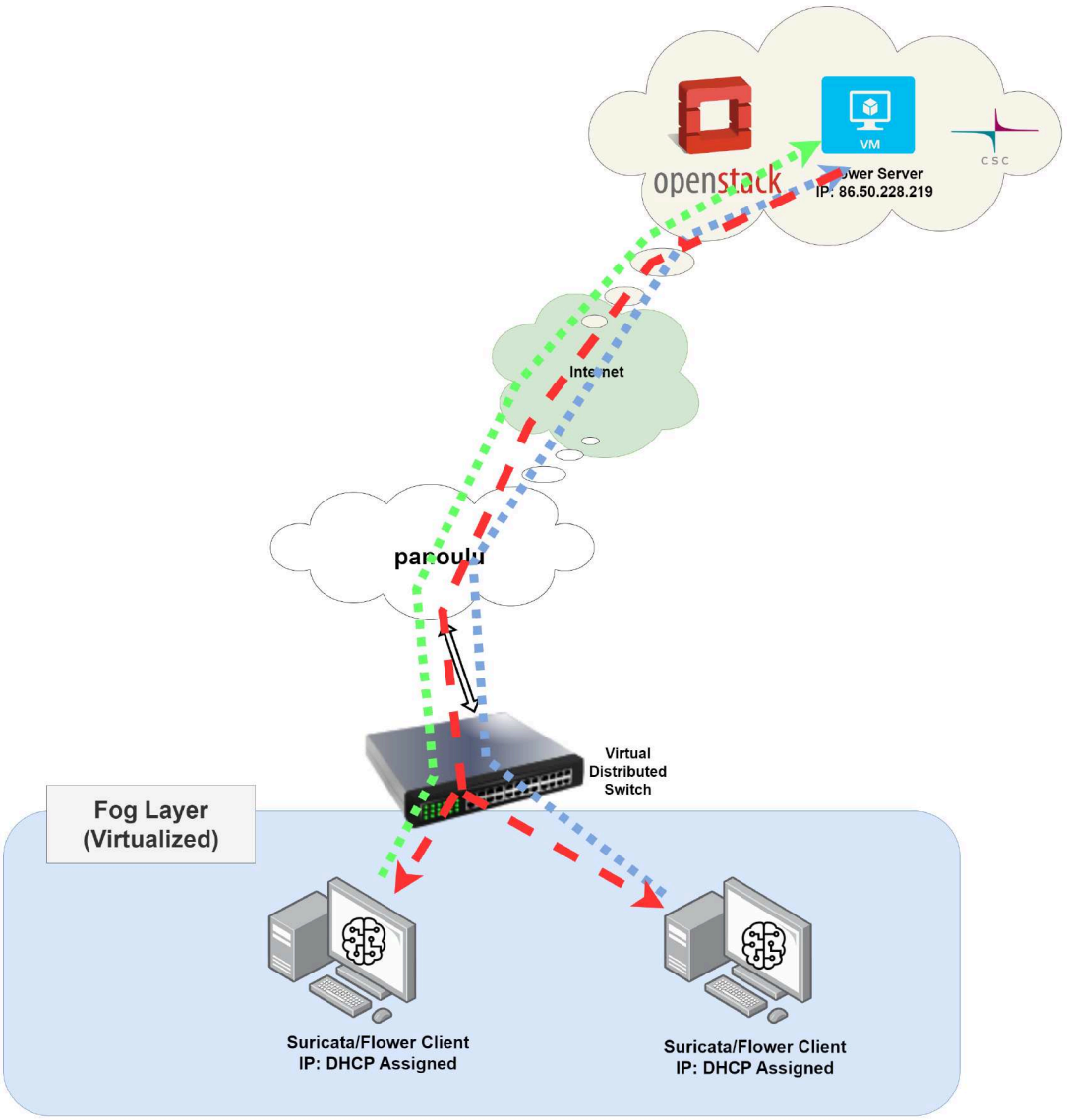

Figure 8. Federated learning illustration in the architecture

FED-ML clients to learn from each other, as each client has access to specific network traffic and not all the traffic.

4.1. Physical Devices

To implement the proposed architecture, we needed actual physical devices that could provide the required computational resources for the whole cyber range. The following is a list of devices that were employed for this purpose.

4.1.1. Computational Devices

The following lists the computational devices used in the proposed architecture.

- Server
  - Processor: AMD Ryzen Threadripper 3960X 24 core
  - RAM: 128 GB
  - Two NICs
- HP EliteBook 850 G3
  - Processor: Inter(R) Core(TM) i7-6600U CPU 2.60 GHz
  - One NIC
  - RAM: 16 GB
- Zotac ZBOX-EN1070
  - Processor: Inter(R) Core(TM) i5-6400 CPU 2.20 GHz
  - Two NICs
  - RAM: 32 GB
- Lenovo ThinkStation 30D0S39X00
  - Processor: Intel(R) Core(TM) i7-9700 CPU 3.00 GHz
  - RAM: 32 GB
  - One NIC
- Raspberry Pi4 Model B Starter Kit (pi-8gb-starterkit)
  - Processor: ARM Limited Cortex-A72 r0p3
  - RAM: 8 GB
  - One NIC

4.1.2. Network Devices

The following lists the network devices employed for the proposed testbed.

- TP-Link TL-SG108E Gigabit Switch 8 ports
- TP-Link Archer MR600 4G+ Cat6 AC1200 Wireless Dual Band Gigabit Router

4.1.3. Network Storage Device

Here is the list of components used for the SAN & NAS device employed in the architecture.

- 1 X Synology Disk Station DS920+ 4BAY 2.0 GHZ QC 2X GBE 4GB
- 2 X Samsung 970 EVO Plus 500GB M-2-NVME SSD
- 1X Synology 4GB SO-DIMM DDR4 2666MHZ NON-ECC
- 4X Synology HAT5310 NAS 8TB SATA/600 HDD

5. RESULTS AND APPLICATIONS

This section covers the usage of the implemented architecture, emphasizing on 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 [60].

5.1. Network Performance Evaluation

To ensure the network devices' capabilities support the experiment in the proposed architecture, the iperf3 tool was used to gauge 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 9, 10, 11, and 12. The plots indicate that the network devices attached to the panOULU have sufficient bandwidth and speed to support the proposed architecture.

5.2. Data Collection

To process network traffic, the first step is to capture network data in pcap format. Two of the collector systems with Suricata software installed on them were used to capture the network traffic in transit through the panOULU network. Suricata is a widely employed open-source software for network analysis and threat detection, extensively utilized by both private and public organizations and incorporated by prominent vendors for safeguarding their assets.

The network traffic flowing in the proposed architecture is recorded by the Suricata-installed systems, which keep track of the systems in the cyber range interacting with each other on the panOULU network while their traffic is captured for further analysis. The traffic is mirrored to these systems through the SPAN port in the switch, enabling them to capture the entire traffic in the cyber range depicted in Figure 7. The network configuration in the systems is set to promiscuous mode, which enables them to listen to the 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), and rotated when the maximum amount of files is reached, with older files being replaced by new ones. 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.

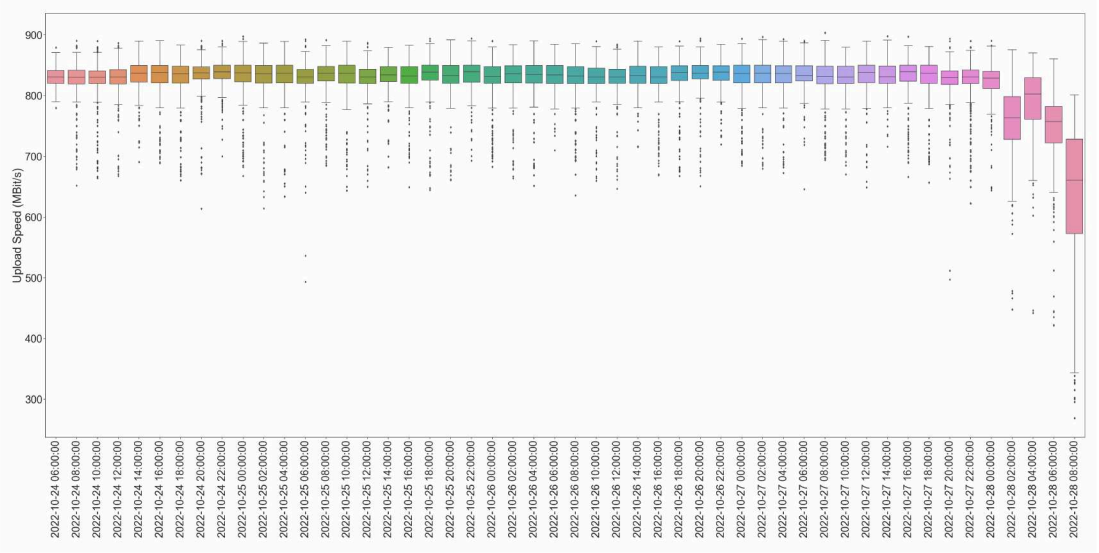

Figure 9. TCP box plot

5.3. 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 presents 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 for 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. Figure 13 displays the outcome of the analysis for the most useful 150 features with fewer null values on a portion of the collected data. The plot illustrates that the number of null values increases as we move towards the less usable features.

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. By choosing these features, the efficiency of the federated machine learning experiment was improved, while still providing enough information to detect potential threats.

Table 2 presents a list of 150 features, along with their description. 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 the categorical features.

The learning phase cycles utilized 37 features as indicated in Table 1. After labeling the batch data, three features were deducted which include source and destination IPs and the epoch time. Hence, the overall feature count was subsequently reduced to 37. Although these features are mainly numerical, they may occasionally contain categorical values. In such cases, a default numerical value was used to replace the categorical value.

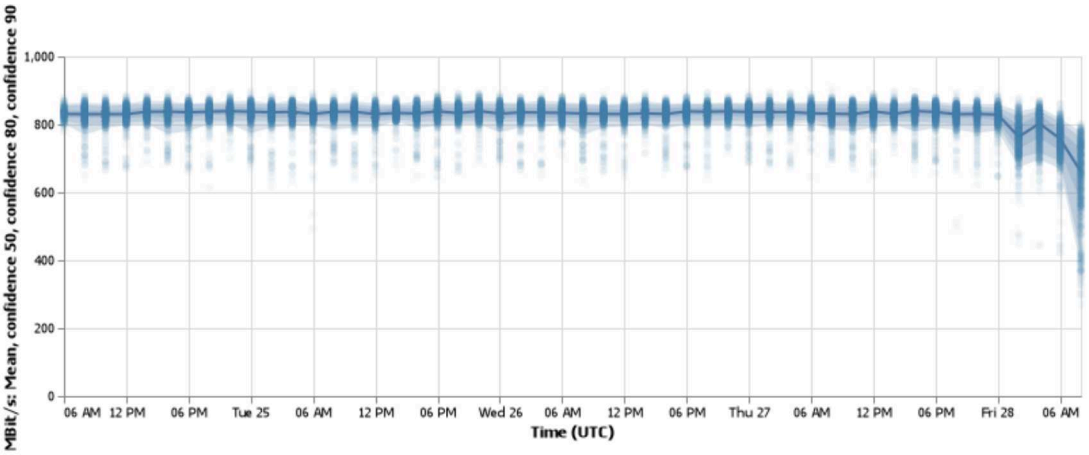

Figure 10. TCP ribbon plot

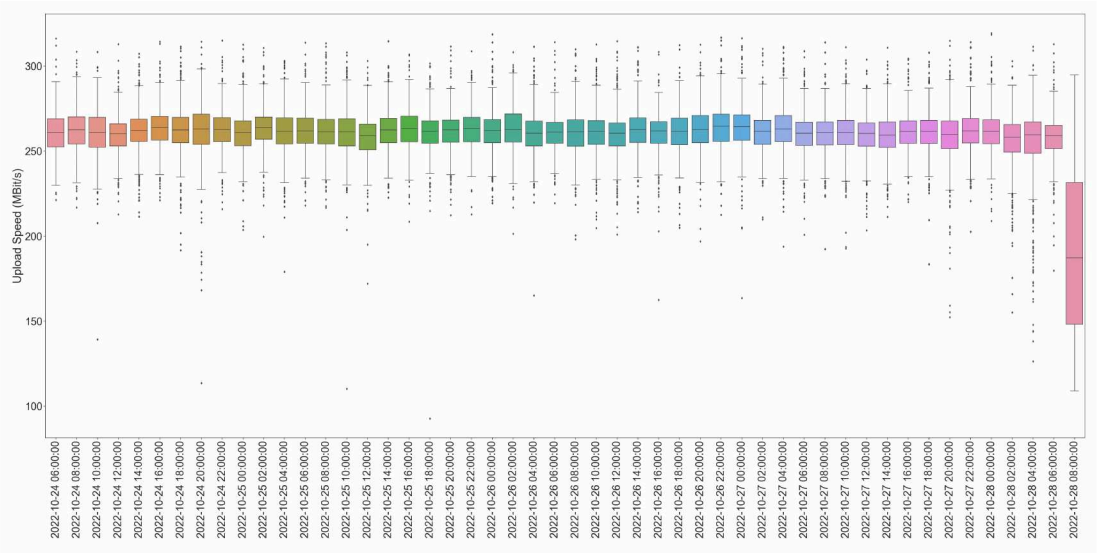

Figure 11. UDP box plot

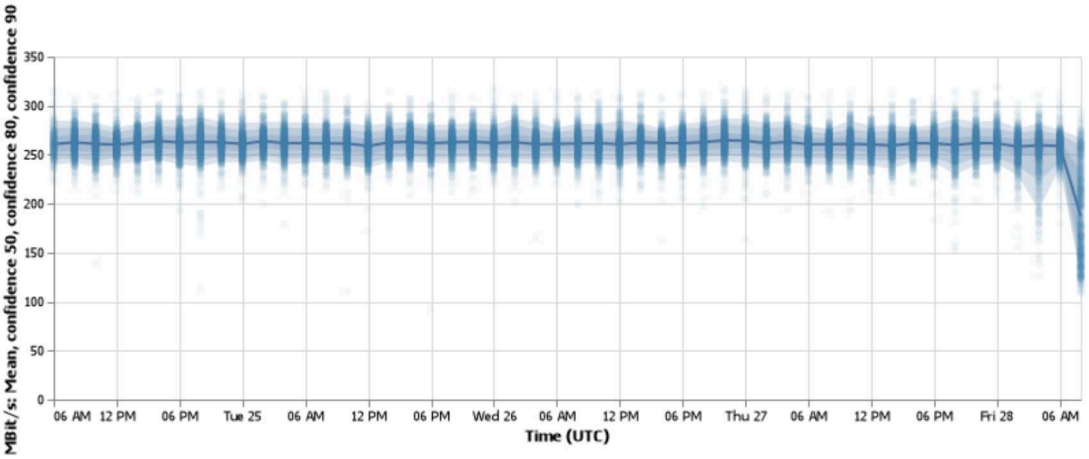

Figure 12. UDP ribbon plot

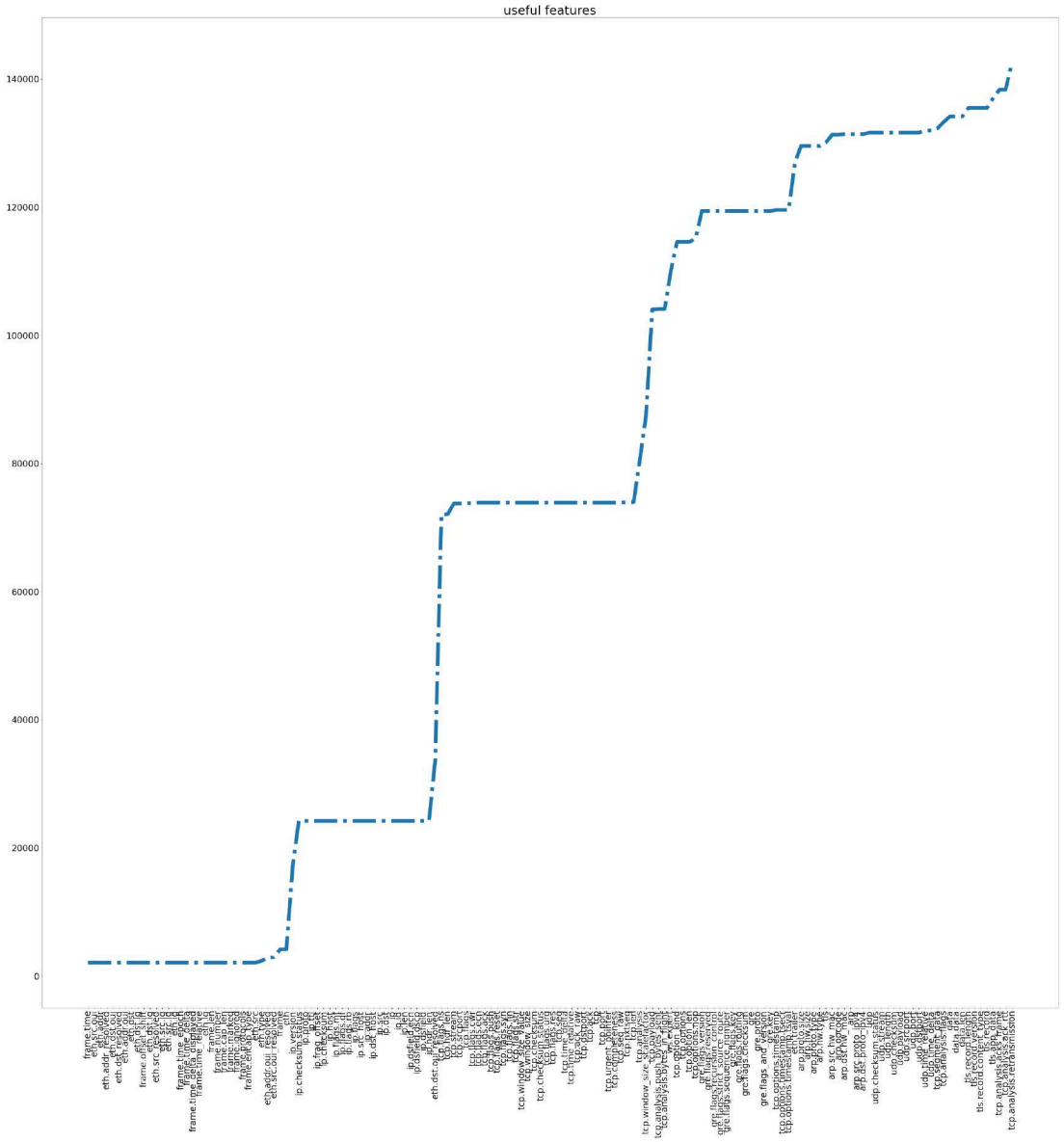

Figure 13. 150 First useful features. The vertical axis displays the count of null values within each feature. Going towards the right in the X axis, the null values tend to increase for these features.

Table 1. This table shows most 37 useful features employed during FED-ML phase with examples.

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

5.4. Labeling the Streaming Data

In order to differentiate between regular network activities and malicious ones in the streaming data, a probing scan is conducted on cyber-range victim VMs within the panOULU network. The scan is performed using hacking 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 attack are categorized as attack packets. In contrast, packets that don't meet these criteria are designated as normal network packets. This classification procedure is essential for machine learning algorithms, which depend on accurately labeled packets to perform classification tasks.

Python is utilized to process the network pcap files that are supplied by suricata-installed machines. The packets are labeled based on their features, which are extracted and converted to CSV format. It should be noted that the whole explained process is entirely automated, with no need for human intervention.

5.5. Data Analytics

The Python scripting language was used in addition to multiple packages for machine learning and deep learning methodologies, as well as other necessary packages for data processing and manipulation to analyze the streaming data. These packages undertake crucial roles in pre-processing, feature selection, and machine learning classification. Some of the packages used for the analysis include 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 behavior 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 on unseen data. To evaluate their performance, 3,000 ML-unseen pcap files were used as mentioned earlier. However, it was not feasible to process all the data simultaneously as it was too large in volume. Despite having a powerful server with 128GB of RAM, we had to devise innovative programming solutions to overcome this challenge.

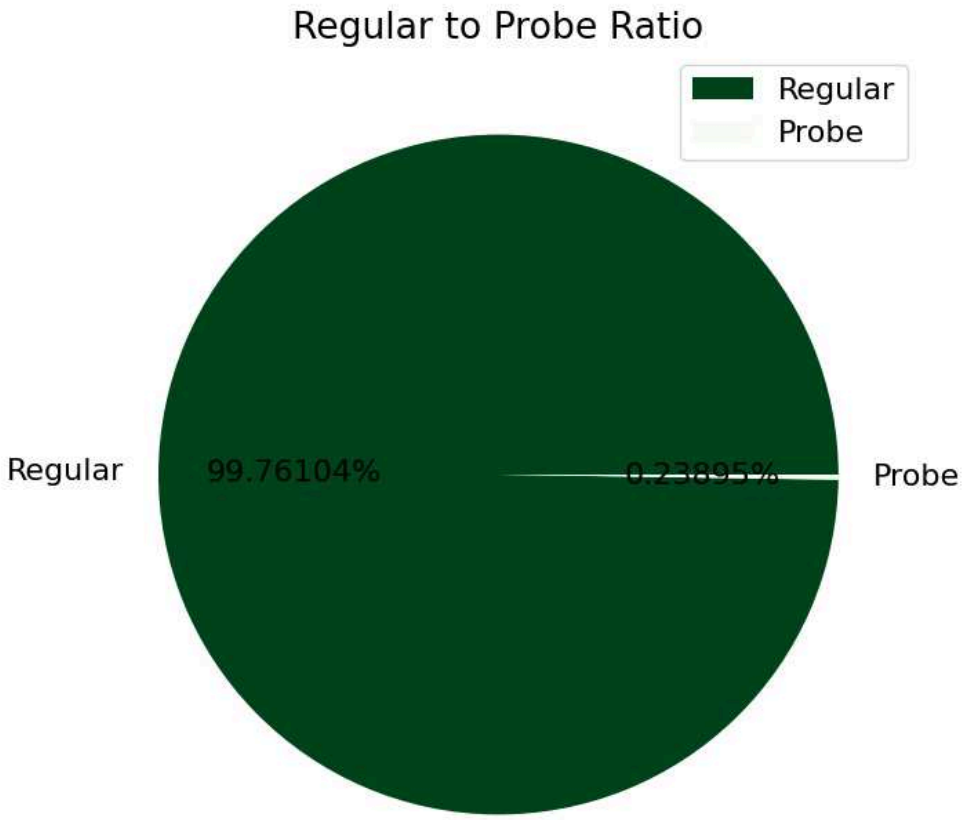

Figure 14. Pie chart for regular packets to probe packets ratio for evaluation data (3000 pcap files).

To cope with the substantial volume of data, 110GB of RAM and 350GB of swap memory were allocated. To expedite the entire process, multi-processing techniques were implemented, which allowed multiple processor cores to be utilized simultaneously.

It should be noted that processing such a large volume of data requires significant computational resources. Even with the most powerful servers available today, it is not possible to process all the data at once. Therefore, novel techniques and programming solutions are required to tackle the challenges associated with processing large datasets.

The ratio between regular network traffic packets and malicious packets for the 3,000 pcap files that were utilized to evaluate the FED-ML trained models is displayed in Figure 14. After extracting packets from these 3,000 pcap files, a total of 2,789,874,382 entries were obtained. Out of this total, 2,783,207,918 packets were regular packets, while the remaining 6,666,464 packets were probe packets.

5.6. 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 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.

During the experiment, two rounds of training were conducted for the FED-ML models. In the initial round, four models were trained in a federated manner for 5 rounds and their performance was evaluated. Since there were no comparable experiments in the literature that was similar to this study, a second round of training was conducted. In the second round, the same four models were trained on streaming data with 50 rounds of training to allow for comparisons and to identify any similarities or differences that arose due to the fluctuations in the quantity of training iterations.

The second round of training was conducted with the goal of comparing the results and identifying 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.

5.6.1. Nuts and Bolts of the Streaming Data

According to Figure 7, the two Suricata machines function as Flower client machines, collecting network activities and performing the role of IDS systems. The streaming data utilized in the FED-ML algorithms is procured separately from these machines. It is fortunate that we have logged all essential steps of the process. As a result, we can examine the logs for these machines and present statistical information about the foundational data employed in the training stage of the machine learning algorithms.

The training process of the four chosen machine learning models commenced on January 17, 2023, at 09:36:28 on the first Suricata machine and ended on January 29, 2023, at 05:18:44. 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 discarded because they did not contain any sign of 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 complain 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).

For the second Suricata machine, training of the models began on January 17, 2023, at 09:36:35, and ended on January 29, 2023, at 05:18:44. 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. This highlights the significant amount of data that was processed and analyzed during the training phase.

Figure 15 provides additional insight into the regular to probe ratio for streaming data for both Flower clients during the training phase.

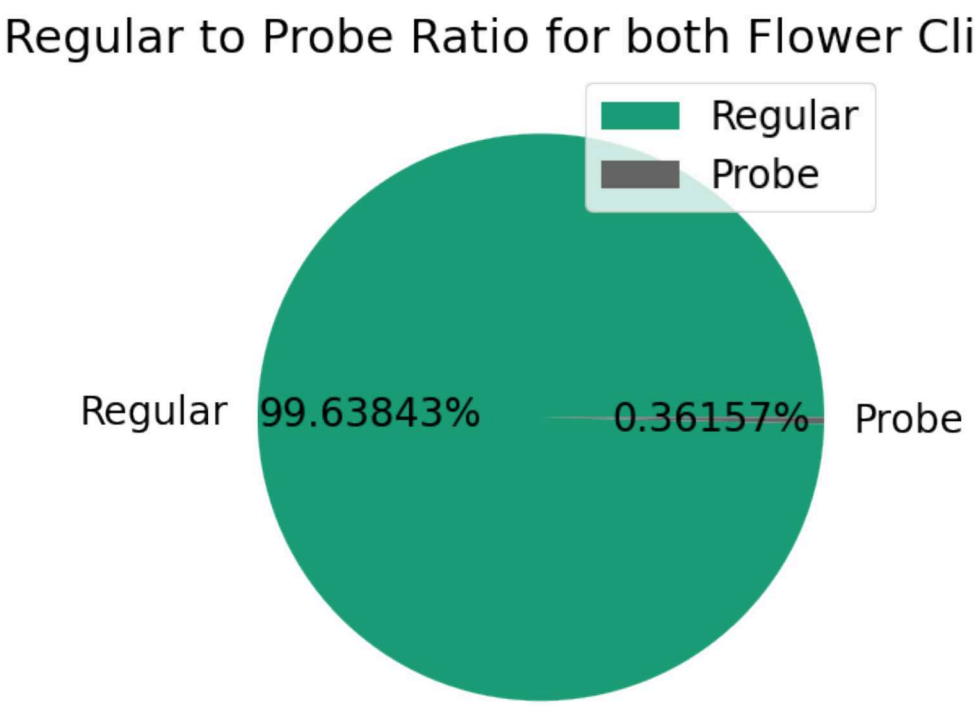

Figure 15. This pie chart depicts the regular packets to probe packets during the training phase for both Flower client machines (suricata machines) for the streaming data.

The following section will discuss the algorithms and technical details. Confusion matrices for models with blue and green colors will be presented for 5 and 50 rounds of training, respectively.

5.6.2. Logistic Regression

Logistic regression is a widely employed statistical model that finds frequent application in classification tasks and predictive analytics. The model predicts the likelihood of an event happening by analyzing a dataset consisting of independent variables. As the dependent variable represents a probability, its values are confined within the range of 0 to 1. The model is similar to linear regression in that it assesses the association between a dependent variable and one or more independent variables. However, logistic regression is utilized for generating predictions concerning categorical variables as opposed to continuous ones. As an illustration, a categorical variable may encompass options such as true or false, yes or no, or 1 or 0.

Logistic regression is categorized as a supervised machine learning model, meaning that it requires labeled data to train the model. Additionally, it is acknowledged as a discriminative model, striving to differentiate between different classes. When it comes to logistic regression, the model attempts to classify an input into one of two classes. The model accomplishes this by computing the ratio of odds for the

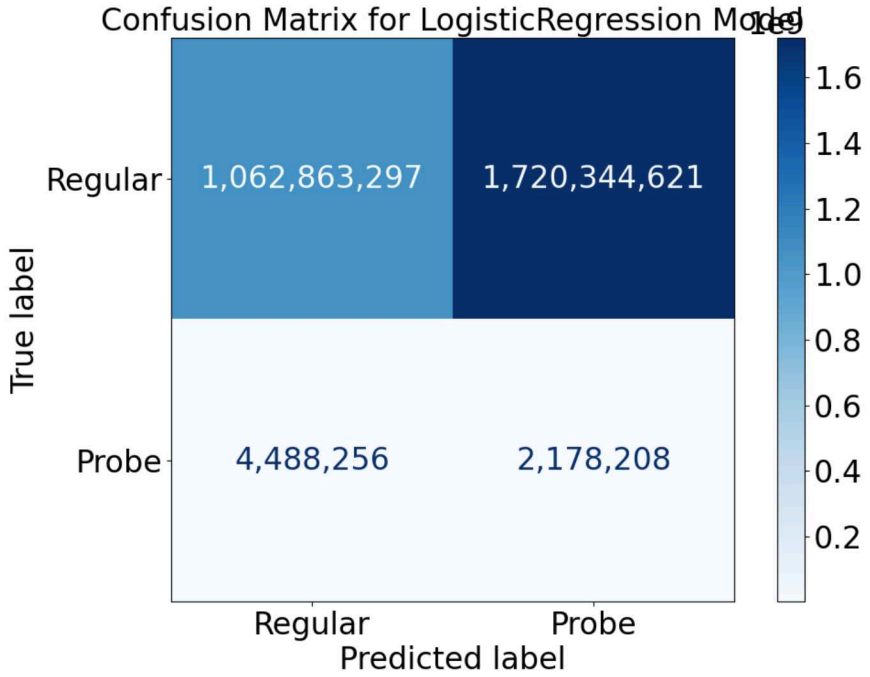

Figure 16. Confusion matrix for logistic regression classifier after 5 rounds of training.

likelihood of an event transpiring, based on a specified set of independent variables. The transformed probability is derived by converting the odds ratio using the logistic function, which maps the odds ratio to a probability value between 0 and 1.

Logistic regression finds extensive application in various domains, such as fraud detection, marketing, and medical diagnosis. It is often used in situations where the outcome is binary, such as predicting whether a customer will make a purchase or not. The model can also be extended to handle more than two classes by using a technique called multinomial logistic regression. However, this requires a different approach to calculating the odds ratio and transforming it into a probability.

A logistic regression classifier from sklearn was employed in the FED-ML architecture to train the model via streaming data in the proposed cyber range test bed. The confusion matrix and evaluation of the unseen data are shown below for both 5 and 50 rounds of training.

As shown in Figure 16, this classifier successfully predicted 2,178,208 probe packets after 5 rounds of training. However, it misclassified 1,720,344,621 of the regular packets as probe packets, while misclassifying 4,488,256 of the probe packets as regular ones. Respectively, the accuracy, precision, recall, and F1-score values are 0.3398, 0.0011, 0.2248, and 0.0021.

The evaluation result from the same FED-ML trained model with 50 rounds of training is shown in Figure 17. The figure depicts that the classifier correctly predicted 232 probe packets. The corresponding metrics for the classification model are as follows: accuracy of 0.9974, precision of 0.0045, recall of 0.0000, and F1-score of 0.0000.

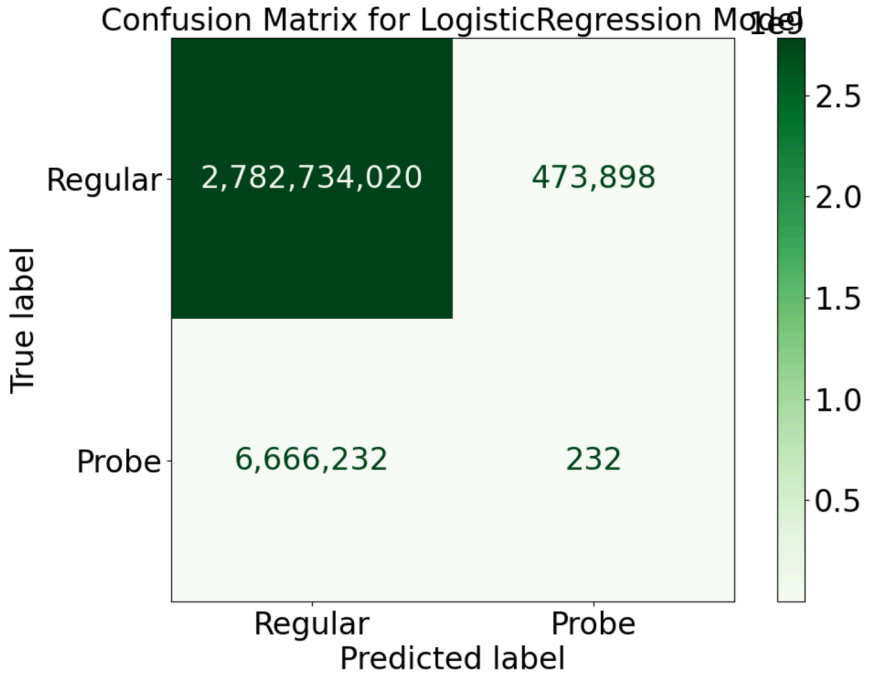

Figure 17. Confusion matrix for logistic regression classifier after 50 rounds of training.

5.6.3. 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.

Using the sklearn package, a Perceptron model was trained in FED-ML architecture to classify packet traffic in the proposed cyber range. The ensuing figures depict the confusion matrix and assessment outcomes of the unseen data. As Figure 18 shows, the Perceptron model trained with 5 rounds could successfully predict 2,684,551 probe packets. However, 1,666,601,760 packets were misclassified as probe and 3,981,913 packets were misclassified as regular. The recorded values for accuracy, precision, recall, and F1-score were 0.3684, 0.0014, 0.4291, and 0.0029, in that order.

The evaluation result of the Perceptron model trained with 50 rounds in FED-ML is shown in Figure 19. The plot shows that the classifier correctly predicted 1,291,882 probe packets. The respective metrics for accuracy, precision, recall, and F1-score were 0.9870, 0.0889, 0.2406, and 0.0897.

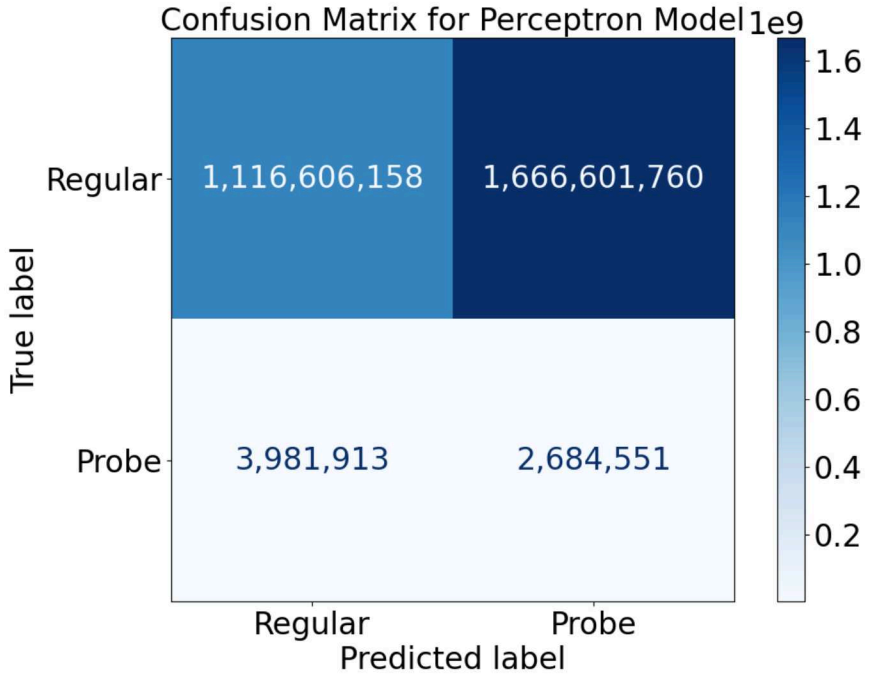

Figure 18. Confusion matrix for perceptron classifier after 5 rounds of training.

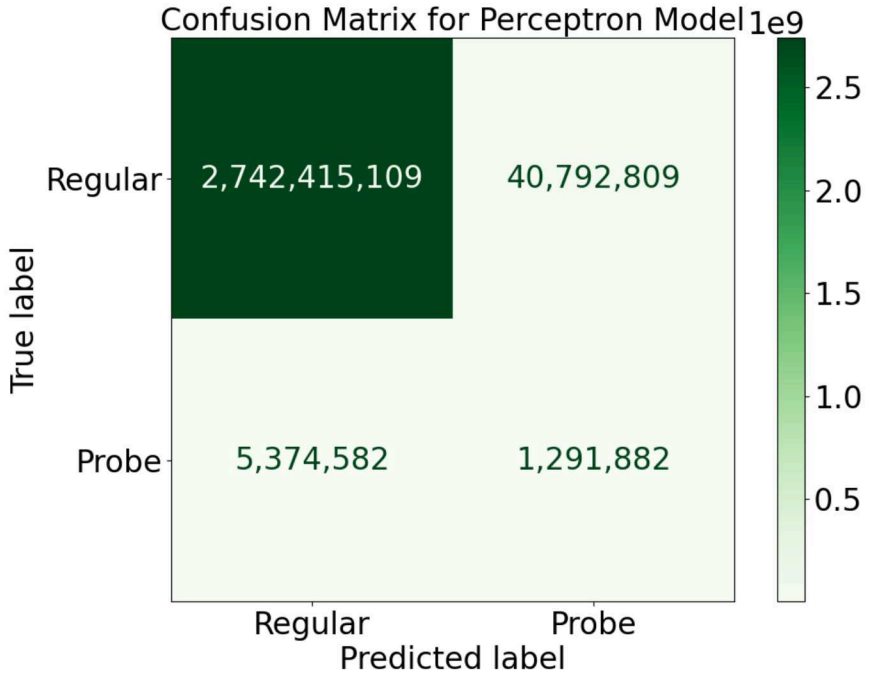

Figure 19. Confusion matrix for perceptron classifier after 50 rounds of training.

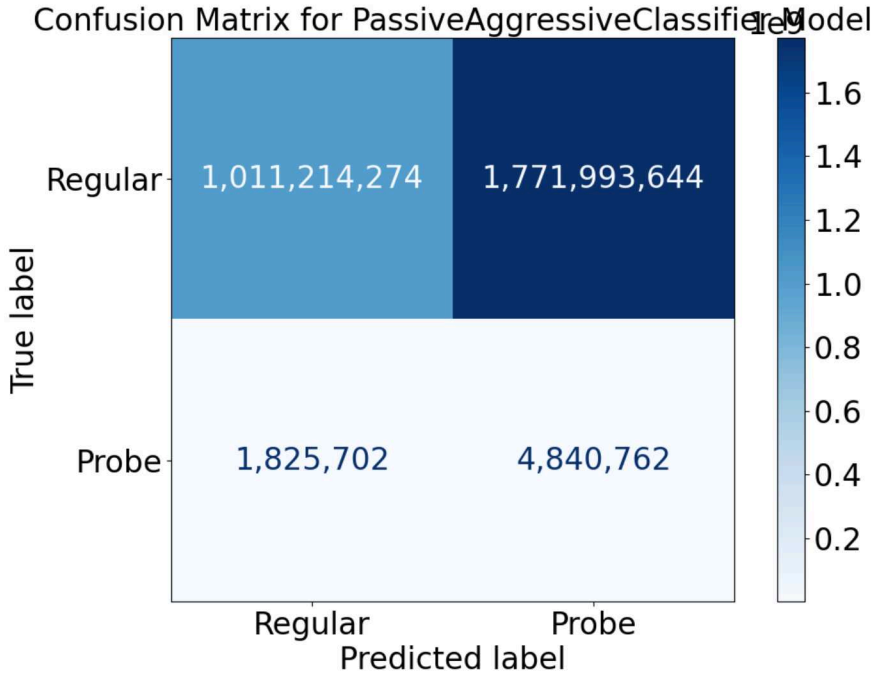

Figure 20. Confusion matrix for passive aggressive classifier after 5 rounds of training.

5.6.4. 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 guaranting 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.

A passive aggressive model was trained in FED-ML architecture using the streaming data in the cyber range by utilizing the sklearn package. The following presents the confusion matrix and evaluation of the unseen data after 5 rounds of training. As depicted in Figure 20, the classifier successfully predicted 4,840,762 probe packets. Misclassifications occurred for regular packets, which were classified as probe packets, and probe packets that were classified as regular ones, with 1,771,993,644 and 1,825,702 misclassifications respectively. The obtained accuracy, precision, recall, and F1-score were 0.3224, 0.0024, 0.6172, and 0.0047, correspondingly.

The evaluation result from the FED-ML trained passive aggressive model with 50 rounds of training is shown in figure 21. The plot illustrates that the classifier was able to correctly predict 942,946 packets. The accuracy, precision, recall, and F1-score are 0.9953, 0.1115, 0.1728, and 0.1005, respectively.

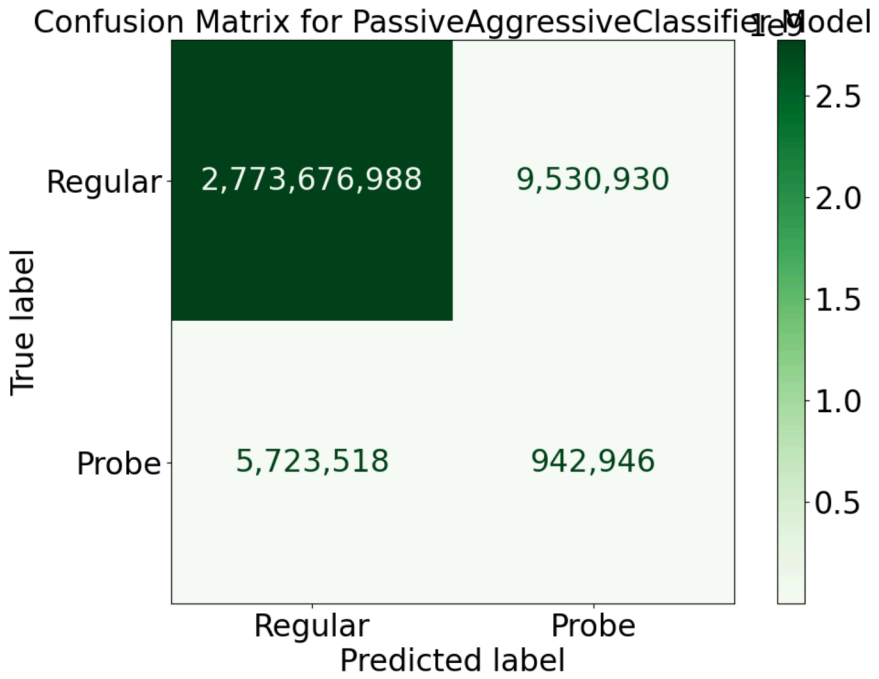

Figure 21. Confusion matrix for passive aggressive classifier after 50 rounds of training.

5.6.5. 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

Scikit-learn offers the SGDClassifier module for implementing SGD classification. Using this classifier, a model was trained in FED-ML architecture using the streaming data in the cyber range. The confusion matrix and evaluation of the unseen data are presented below. As depicted in figure 22, the classifier successfully predicted 72,912 probe packets after 5 rounds of training. However, 1,678,441,755 regular packets were misclassified as probe and 6,593,552 probe packets were misclassified as regular. The classifier achieves an accuracy of 0.3616, with precision, recall, and F1-score values of 0.0000, 0.0353, and 0.0000, respectively.

The evaluation results from the same FED-ML trained model with 50 rounds of training can be seen in figure 23. The plot shows that the classifier correctly predicted 533,391 probe packets. The accuracy, precision, recall, and F1-score for the model were 0.9902, 0.0460, 0.1189, and 0.1189, respectively.

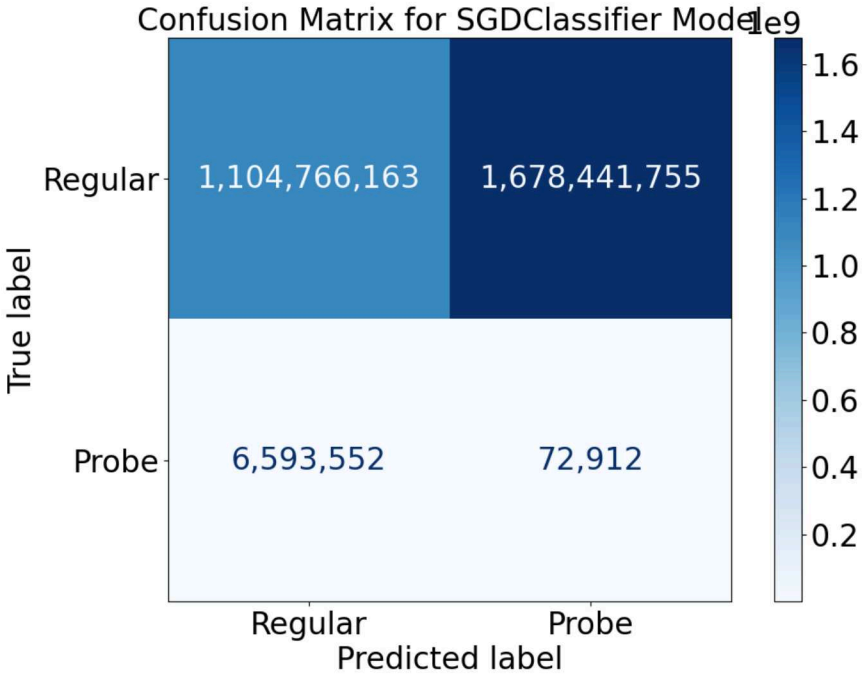

Figure 22. Confusion matrix for SGD classifier after 5 rounds of training.

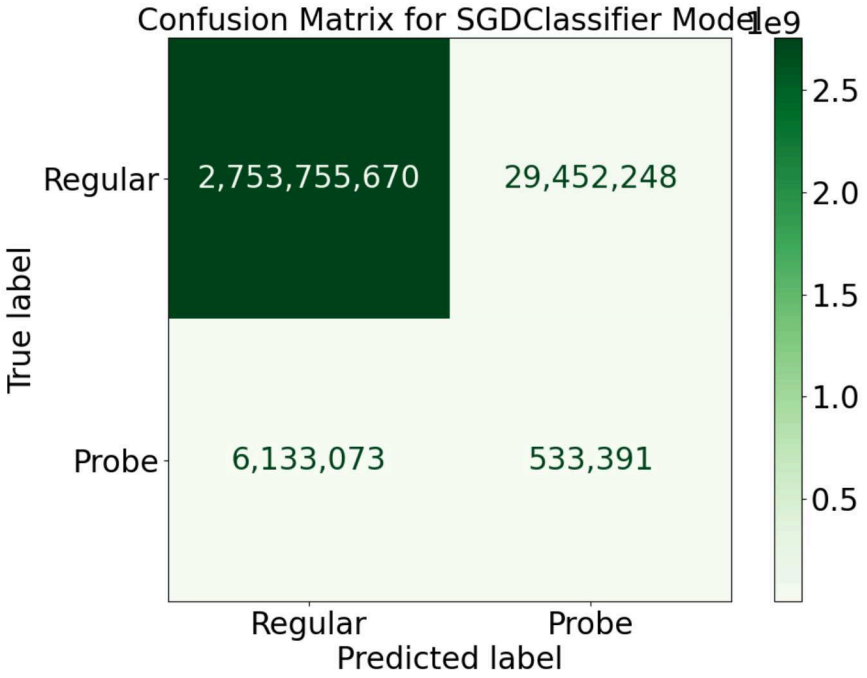

Figure 23. Confusion matrix for SGD classifier after 50 rounds of training.

6. DISCUSSION

The creation of AI-based intrusion detection systems requires the application of high quality data to train the models. A wealth of datasets has been publicly disclosed over past years to facilitate researchers in developing high-performance models. Notwithstanding, a large portion of the introduced architectures and datasets has become obsolete. Consequently, the central objective of this thesis is to engineer a contemporary testbed architecture suitable for defense systems, factoring in the ceaseless evolution of technologies and streaming Federated Learning models. Below we discuss the strengths and limitations of the proposed solutions.

In the research conducted, the majority of predictions were negative, with only a small fraction being positive. As a result, unsatisfactory results were obtained, 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. The usage of IP addresses can potentially introduce biased results, and disregarding them can lead to inaccurate outcomes. It is crucial to recognize the limitations of not using IP addresses, including the distribution of cases.

The developed testbed is unique compared to other testbeds discussed in the prior research section (2) 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 being labeled as they flow in the network, enabling data collection and pre-processing to occur concurrently with the learning phase of FED-ML models.

In contrast to existing literature, which typically follows a sequential approach of collecting data first and then performing pre-processing for ML models, this method introduces a groundbreaking approach. In this traditional approach, there is no concurrency, and there are multiple interruptions in the whole pipeline that require manual human involvement. In contrast, our approach enables the employed models to capture the dynamism of the network flow on-the-go.

It is worth emphasizing that the models employed in this study have been trained using streaming data, meaning that they have seen different data streams during their training phase. This approach is nonconformist compared to traditional machine learning models examined in the earlier section of research, which are often trained using the same set of data. However, since the models in this work see a lot of data during their learning time, the discrepancy of their learning data shouldn't make much of a difference over the long run.

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 method of evaluating the models aligns with other studies in the field where the models are also tested 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 [61], it was not possible to perform this step in the current study due to the streaming nature of the data. Previous works, such as [62], [10], and [63], have utilized normalization techniques in their work. However, in the proposed architecture of this study, each round during the learning phase could be viewed as a new opportunity for the machine learning models to be appropriately trained, and normalization could still be applied.

The effectiveness of the selected features may not have been as expected, and there could be other features that would benefit 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. Unlike the conventional approach taken by other related works of training models, where all data is available during the learning phase, streaming data is not readily available all at once.

The reason why it is not possible to transform categorical data into numerical equivalents is because there is always a possibility that a new category could appear in the future data. Converting categorical values into numerical equivalents is a common practice in related works, such as [64]. However, this requirement poses a challenge for the federated learning case, which deals with streaming data. One of the findings of this work is the need to address this challenge.

Plot 14 indicates a notable disparity between the quantity of regular packets and probe packets, which may have adverse effects on the efficacy of machine learning algorithms trained on this data. A balanced dataset is crucial for optimal model performance, with each category represented 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 network or its users.

Other related studies such as [65], [66] and [67] 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.

The use of packet-level features alone in this work may not be sufficient for accurately identifying malicious activity 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 have a positive impact on 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 [47], these features were included.

6.1. Answers to Research Questions

In the introduction sections we laid out four fundamental research questions for this work. Now that we have gone through all the works and the final results, we can answer each of these questions in an explicit manner as the following:

- **Answer to Research Question 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 helped create a single virtual switch that spans multiple physical hosts, enabling centralized network management and configuration, which could were further utilized for capturing and analyzing PCAP files.

  Furthermore, the service orchestration feature provided by the vCenter Server Appliance enabled integrated control of the devices in the fog layer. By using this feature, the process of deploying, managing, and monitoring virtual machines in the cyber range was streamlined.

  Moreover, the federated learning architecture of the work enabled the use of edge/cloud computing, as the Suricata-installed devices were responsible for training the models locally and sending the updated results back to the cloud. This had the benefit of reducing latency and providing a cost-effective way of storing and managing large volumes of data.

- **Answer to Research Question 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 model.

  This paradigm could be taken advantage of by extending the leverage of federation to different layers in any other situation. This level of granularity means that even edge devices, like Raspberry Pi, for instance, could train models locally without needing access to the entire network traffic. They could simply train the models locally and send updates back to the Flower server.

- **Answer to Research Question 3:** The proposed architecture aimed to showcase the intricate nature of modern-day network topologies. The setup includes both wired and wireless connections to the panOULU network, which encompasses diverse topologies. The devices used in this work span across IoT devices, smartphones, VMs, and more.

By combining these devices with various communication media and technologies, a realistic testbed has been established. This testbed includes different tiers such as edge, fog, and cloud, which can be utilized to generate a high-quality 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, it has become crucial to develop such a testbed that accurately replicates real-world scenarios.

- **Answer to Research Question 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 plots and figures generated from the trained machine learning models clearly demonstrate 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 a real world.

  It could be inferred from the results that the best possible way to protect the network assets and devices is by securing them in the first place by setting up strict firewall rules and limiting the inbound and outbound activities of these devices. Keeping the operating systems of these devices up-to-date is another precaution that could mitigate the security breach of these devices.

6.2. Limitations and Future Work

A selection of ML algorithms was chosen to evaluate the results in this study. 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 the proposed architecture. 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 resulting in a more balanced dataset compared to the current status of the data ratio as illustrated in figure 14.

In addition, it would be interesting and necessary to extend this work by performing different types of attacks on the cyber range and training ML models on these diverse attacks. The ML models utilized in this work have the capability of predicting multi-class data, making it easier to expand this study to multi-attack-type scenarios for FED-ML learning.

Moreover, it is possible to deploy multiple honeypots in various ESXi hosts in a distributed manner to attract attackers engaging in malicious activities within the panOULU network. The resulting data can be utilized to train ML algorithms for cyber security purposes. The proposed architecture can provide the necessary platform for implementing the described blueprint.

7. CONCLUSION

The presented testbed architecture highlights the necessity of a robust and efficient communication infrastructure to facilitate the growing network of interconnected devices, encompassing sensors and IoT devices, cellphones, and drones that are being connected to public networks and the Internet. The future of mobile networks is undergoing a transformative phase beyond the limitations of 4G and forthcoming 5G networks, with enabling technologies paving the way for novel system architectures. However, the cybersecurity of these emerging systems is neglected and fails to keep pace with the necessity for a unified cybersecurity framework. Cybersecurity should be prioritized as critical national infrastructures are targeted by cyber attacks, and a robust Intrusion Detection System (IDS) is a must for attack classification and detection, intrusions, and violations of security policies promptly. The challenges faced by researchers in obtaining valid datasets to evaluate their proposed techniques due to the complexity of networks and systems and the lack of high-quality testbeds were also discussed in this work.

An architecture was outlined and implemented to facilitate the deployment of Federated Machine Learning (FED-ML) algorithms designed for real-time analysis of network traffic to enhance cybersecurity. The experimental results of the deployed FED-ML model were discussed, which was trained on streaming data flowing through the distributed switch across various layers of the architecture. The feature extraction process involved using tshark to extract features from binary pcap files, with Python serving as the underlying language for the entire work. The streaming data was classified into normal network activities and malicious ones by performing a probing scan on the cyber-range victim VMs, and the process was described.

The fog layer consisted of key components such as VMware vSphere hypervisor, Service Orchestration (SO), SDN, NFV, as mentioned. Streaming data was utilized to train the models, and normalization was not conducted while learning. Upon analyzing the results, it was observed that the chosen features may not be as efficient as anticipated, and the regular to probe packet ratio was imbalanced. Some of the possible reasons for such results were further outlined in the discussion section (6).

8. REFERENCES

[1] Taleb T., ika Benza C., Lopez M.B., Mikhaylov K., Tarkoma S., Kostakos P., Mahmood N.H., Pirinen P., Matinmikko-Blue M., Latva-aho M. et al. (2022) 6g system architecture: A service of services vision. ITU journal on future and evolving technologies 3, pp. 710–743.

[2] Alcácer V. & Cruz-Machado V. (2019) Scanning the industry 4.0: A literature review on technologies for manufacturing systems. Engineering Science and Technology, an International Journal 22, pp. 899–919. URL: https://www.sciencedirect.com/science/article/pii/S2215098618317750.

[3] O'Donovan P., Gallagher C., Bruton K. & O'Sullivan D.T. (2018) A fog computing industrial cyber-physical system for embedded low-latency machine learning industry 4.0 applications. Manufacturing Letters 15, pp. 139–142. URL: https://www.sciencedirect.com/science/article/pii/S2213846318300087, industry 4.0 and Smart Manufacturing.

[4] Ferrag M.A., Maglaras L., Moschoyiannis S. & Janicke H. (2020) Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications 50, p. 102419. URL: https://www.sciencedirect.com/science/article/pii/S2214212619305046.

[5] Maglaras L.A., Kim K.H., Janicke H., Ferrag M.A., Rallis S., Fragkou P., Maglaras A. & Cruz T.J. (2018) Cyber security of critical infrastructures. ICT Express 4, pp. 42–45. URL: https://www.sciencedirect.com/science/article/pii/S2405959517303880, sI: CI & Smart Grid Cyber Security.

[6] Mukherjee B., Heberlein L. & Levitt K. (1994) Network intrusion detection. IEEE Network 8, pp. 26–41.

[7] Larson D. (2016) Distributed denial of service attacks - holding back the flood. Netw. Secur. 2016, p. 5–7. URL: https://doi.org/10.1016/S1353-4858(16)30026-5.

[8] Venkatraman S. & Alazab M. (2018) Use of data visualisation for zero-day malware detection. Security and Communication Networks 2018, pp. 1–14.

[9] Mishra P., Varadharajan V., Tupakula U. & Pilli E.S. (2019) A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials 21, pp. 686–728.

[10] Vinayakumar R., Alazab M., Soman K.P., Poornachandran P., Al-Nemrat A. & Venkatraman S. (2019) Deep learning approach for intelligent intrusion detection system. IEEE Access 7, pp. 41525–41550.

[11] Paxson V. (1999) Bro: a system for detecting network intruders in real-time. Computer Networks 31, pp. 2435–2463. URL: https://www.sciencedirect.com/science/article/pii/S1389128699001127.

[12] LeCun Y., Bengio Y. & Hinton G. (2015) Deep learning. Nature 521, pp. 436–44.

[13] Venkatraman S. & Alazab M. (2018) Use of data visualisation for zero-day malware detection. Security and Communication Networks 2018, pp. 1–13.

[14] Lee W. & Stolfo S.J. (2000) A framework for constructing features and models for intrusion detection systems. ACM Trans. Inf. Syst. Secur. 3, p. 227–261. URL: https://doi.org/10.1145/382912.382914.

[15] Özgür A. & Erdem H. (2016), A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015.

[16] Agarwal R.C. & Joshi M.V. (2001) Pnrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection). In: SDM.

[17] Kayacik H.G., Zincir-Heywood A.N. & Heywood M.I. (2005) Selecting features for intrusion detection: A feature relevance analysis on kdd 99. In: Conference on Privacy, Security and Trust.

[18] Zhang J., Zulkernine M. & Haque A. (2008) Random-forests-based network intrusion detection systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38, pp. 649–659.

[19] Huda S., Abawajy J., Alazab M., Abdollalihian M., Islam M.R. & Yearwood J. (2014) Hybrids of support vector machine wrapper and filter based framework for malware detection. Future Generation Computer Systems 55.

[20] Alazab M., Huda S., Abawajy J., Islam M.R., Yearwood J., Venkatraman S. & Broadhurst R. (2014) A hybrid wrapper-filter approach for malware detection. Journal of Networks 9.

[21] Hu W., Hu W. & Maybank S. (2008) Adaboost-based algorithm for network intrusion detection. IEEE Transactions on Systems, Man, and Cybernetics, Part B 38, pp. 577–583.

[22] Ertöz L., Steinbach M. & Kumar V. (2003) Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data.

[23] Amor N.B., Benferhat S. & Elouedi Z. (2004) Naive bayes vs decision trees in intrusion detection systems. In: ACM Symposium on Applied Computing.

[24] Valdes A. & Skinner K. (2000) Adaptive, model-based monitoring for cyber attack detection. pp. 80–92.

[25] Yeung D.Y. & Chow C. (2002) Parzen-window network intrusion detectors .

[26] Li W. (2004) Using genetic algorithm for network intrusion detection .

[27] Kolias C., Kambourakis G. & Maragoudakis M. (2011) Swarm intelligence in intrusion detection: A survey. Computers & Security 30, pp. 625–642. URL: https://www.sciencedirect.com/science/article/pii/S016740481100109X.

[28] Nehinbe J.O. (2011) A critical evaluation of datasets for investigating IDSs and IPSs researches. In: 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS), IEEE. URL: https://doi.org/10.1109/cis.2011.6169141.

[29] Sharafaldin I., Lashkari A.H. & Ghorbani A.A. (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy, SCITEPRESS - Science and Technology Publications. URL: https://doi.org/10.5220/0006639801080116.

[30] Gharib A., Sharafaldin I., Lashkari A.H. & Ghorbani A.A. (2016) An evaluation framework for intrusion detection dataset. In: 2016 International Conference on Information Science and Security (ICISS), IEEE. URL: https://doi.org/10.1109/icissec.2016.7885840.

[31] Sharafaldin I., Habibi Lashkari A. & Ghorbani A.A. (2019) A detailed analysis of the cicids2017 data set. In: P. Mori, S. Furnell & O. Camp (eds.) Information Systems Security and Privacy, Springer International Publishing, Cham, pp. 172–188.

[32] Bay S. (1999), The uci kdd archive. https://kdd.ics.uci.edu/. Irvine, CA: University of California, Department of Computer Science.

[33] Li D., Deng L., Lee M. & Wang H. (2019) Iot data feature extraction and intrusion detection system for smart cities based on deep migration learning. International Journal of Information Management 49, pp. 533–545. URL: https://www.sciencedirect.com/science/article/pii/S0268401218311356.

[34] Song J., Takakura H., Okabe Y., Eto M., Inoue D. & Nakao K. (2011), Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. Paper presented at proceedings of the 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security. Salzburg Austria.

[35] Tavallaee M., Bagheri E., Lu W. & Ghorbani A.A. (2009) A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6.

[36] Tavallaee M., Bagheri E., Lu W. & Ghorbani A. (2009), A detailed analysis of the kdd cup 99 data set. Paper presented at proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications. Ottawa.

[37] Moustafa N. & Slay J. (2016) The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective 25, pp. 18–31. URL: https://doi.org/10.1080/19393555.2015.1125974.

[38] Moustafa N., Slay J. & Creech G. (2019) Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. IEEE Transactions on Big Data 5, pp. 481–494.

[39] Sharafaldin I., Lashkari A.H. & Ghorbani A.A. (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP.

[40] Abdulhammed R., Musafer H., Alessa A., Faezipour M. & Abuzneid A. (2019) Features dimensionality reduction approaches for machine learning based network intrusion detection. Electronics 8, p. 322. URL: https://doi.org/10.3390/electronics8030322.

[41] Sharafaldin I., Lashkari A. & Ghorbani A. (2018), Toward generating a new intrusion detection dataset and intrusion traffic characterization. Paper presented at proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP). Madeira.

[42] Homoliak I., Barabas M., Chmelar P., Drozd M. & Hanacek P. (2013) The steering committee of the world congress in computer science, computer âQ. Proceedings of the International Conference on Security and Management (SAM) , p. 1URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85127925725&partnerID=40&md5=3588bf0d65255264ce9949de9a67d063, cited by: 1.

[43] Jazi H.H., Gonzalez H., Stakhanova N. & Ghorbani A.A. (2017) Detecting http-based application layer dos attacks on web servers in the presence of sampling. Computer Networks 121, pp. 25–36. URL: https://www.sciencedirect.com/science/article/pii/S1389128617301172.

[44] Shiravi A., Shiravi H., Tavallaee M. & Ghorbani A.A. (2012) Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security 31, pp. 357–374. URL: https://www.sciencedirect.com/science/article/pii/S0167404811001672.

[45] Creech G. & Hu J.J.I.T.o.C. (2013) A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns 63, p. 807 – 819. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85093168076&partnerID=40&md5=ee02796b7d947729a2467c2cc34dd01e, cited by: 0.

[46] Creech G. & Hu J. (2013) Generation of a new ids test dataset: Time to retire the kdd collection. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC), pp. 4487–4492.

[47] Moustafa N. (2021) A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. Sustainable Cities and Society 72, p. 102994.

[48] Sabhnani M. & Serpen G. (2004) Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. Intell. Data Anal. 8, p. 403–415.

[49] Mtimet J. & Amiri H. (2013) Image classification using statistical learning for automatic archiving system. International Review on Computers and Software (IRECOS) 8. URL: https://www.praiseworthyprize.org/jsm/index.php?journal=irecos&page=article&op=view&path%5B%5D=11048.

[50] Moustafa N. & Slay J. (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6.

[51] Suricata. URL: https://suricata.io/.

[52] Tiensyrjä J., Ojala T., Hakanen T. & Salmi O. (2010) panoulu conqueror: pervasive location-aware multiplayer game for city-wide wireless network. In: Proceedings of the 3rd International Conference on Fun and Games, pp. 157–165.

[53] Kukka H., Ojala T., Tiensyrjä J. & Mikkonen T. (2008) panoulu luotsi: A location based information mash-up with xml aggregator and wifi positioning. In: Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia, pp. 80–83.

[54] Ojala T., Orajärvi J., Puhakka K., Heikkinen I. & Heikka J. (2011) panoulu: Triple helix driven municipal wireless network providing open and free internet access. In: Proceedings of the 5th International Conference on Communities and Technologies, pp. 118–127.

[55] ESXi installation link. URL: https://customerconnect.vmware.com/en/evalcenter?p=free-esxi7.

[56] Flower official link. URL: https://flower.dev/.

[57] Li C.S. & Liao W. (2013) Software defined networks. IEEE Communications Magazine 51, pp. 113–113.

[58] Han B., Gopalakrishnan V., Ji L. & Lee S. (2015) Network function virtualization: Challenges and opportunities for innovations. IEEE communications magazine 53, pp. 90–97.

[59] WAZUH dashboard ref. link. URL: https://documentation.wazuh.com/current/user-manual/agents/listing/wazuh-dashboard.html.

[60] Evaluation dataset of this work. URL: https://doi.org/10.5281/zenodo.7956304.

[61] Ahmad Z., Shahid Khan A., Wai Shiang C., Abdullah J. & Ahmad F. (2021) Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies 32, p. e4150.

[62] Kelli V., Argyriou V., Lagkas T., Fragulis G., Grigoriou E. & Sarigiannidis P. (2021) Ids for industrial applications: A federated learning approach with active personalization. Sensors 21, p. 6743.

[63] Li S., Qi Q., Wang J., Sun H., Li Y. & Yu F.R. (2020) Ggs: General gradient sparsification for federated learning in edge computing. In: ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE, pp. 1–7.

[64] Mirzaee P.H., Shojafar M., Pooranian Z., Asefy P., Cruickshank H. & Tafazolli R. (2021) Fids: A federated intrusion detection system for 5g smart metering network. In: 2021 17th International Conference on Mobility, Sensing and Networking (MSN), IEEE, pp. 215–222.

[65] Karatas G., Demir O. & Sahingoz O.K. (2020) Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. IEEE access 8, pp. 32150–32162.

[66] Khan F.A., Gumaei A., Derhab A. & Hussain A. (2019) A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access 7, pp. 30373–30385.

[67] Jiang K., Wang W., Wang A. & Wu H. (2020) Network intrusion detection combined hybrid sampling with deep hierarchical network. IEEE Access 8, pp. 32464–32476.

9. APPENDICES

Feature	Description
arp	Address Resolution Protocol
arp.hw.type	Hardware type
arp.proto.type	Protocol type
arp.hw.size	Hardware size
arp.proto.size	Protocol size
arp.opcode	Opcode
arp.src.hw_mac	Sender MAC address
arp.src.proto_ipv4	Sender IP address
arp.dst.hw_mac	Target MAC address
arp.dst.proto_ipv4	Target IP address
data	Data
data.data	Data
data.len	Length
eth	Ethernet
eth.dst	Destination
eth.dst_resolved	Destination (resolved)
eth.dst.oui	Destination OUI
eth.dst.oui_resolved	Destination OUI (resolved)
eth.src	Source
eth.src_resolved	Source (resolved)
eth.src.oui	Source OUI
eth.src.oui_resolved	Source OUI (resolved)
eth.type	Type
eth.addr	Address
eth.addr_resolved	Address (resolved)
eth.addr.oui	Address OUI
eth.addr.oui_resolved	Address OUI (resolved)
eth.trailer	Trailer
eth.dst.lg	LG bit
eth.dst.ig	IG bit
eth.src.lg	LG bit
eth.src.ig	IG bit
eth.lg	LG bit
eth.ig	IG bit
frame	Frame
frame.time	Arrival Time
frame.offset_shift	Time shift for this packet
frame.time_epoch	Epoch Time
frame.time_delta	Time delta from previous captured frame
frame.time_delta_displayed	Time delta from previous displayed frame
frame.time_relative	Time since reference or first frame
frame.number	Frame Number

frame.len	Frame length on the wire
frame.cap_len	Frame length stored into the capture file
frame.marked	Frame is marked
frame.ignored	Frame is ignored
frame.protocols	Protocols in frame
frame.encap_type	Encapsulation type
gre	Generic Routing Encapsulation
gre.proto	Protocol Type
gre.flags_and_version	Flags and Version
gre.flags.checksum	Checksum Bit
gre.flags.routing	Routing Bit
gre.flags.key	Key Bit
gre.flags.sequence_number	Sequence Number Bit
gre.flags.strict_source_route	Strict Source Route Bit
gre.flags.recursion_control	Recursion control
gre.flags.reserved	Flags (Reserved)
gre.flags.version	Version
gre.key	Key
ip	Internet Protocol Version 4
ip.version	Version
ip.hdr_len	Header Length
ip.dsfield	Differentiated Services Field
ip.dsfield.dscp	Differentiated Services Codepoint
ip.dsfield.ecn	Explicit Congestion Notification
ip.len	Total Length
ip.id	Identification
ip.dst	Destination Address
ip.dst_host	Destination Host
ip.src	Source Address
ip.src_host	Source Host
ip.addr	Source or Destination Address
ip.host	Source or Destination Host
ip.flags	Flags
ip.flags.rb	Reserved bit
ip.flags.df	Don’t fragment
ip.flags.mf	More fragments
ip.frag_offset	Fragment Offset
ip.ttl	Time to Live
ip.proto	Protocol
ip.checksum	Header Checksum
ip.checksum.status	Header checksum status
tcp	Transmission Control Protocol
tcp.srcport	Source Port
tcp.dstport	Destination Port
tcp.port	Source or Destination Port

tcp.stream	Stream index
tcp.completeness	Conversation completeness
tcp.seq	Sequence Number
tcp.seq_raw	Sequence Number (raw)
tcp.nxtseq	Next Sequence Number
tcp.ack	Acknowledgment Number
tcp.ack_raw	Acknowledgment number (raw)
tcp.hdr_len	Header Length
tcp.flags	Flags
tcp.flags.res	Reserved
tcp.flags.ns	Nonce
tcp.flags.cwr	Congestion Window Reduced (CWR)
tcp.flags.ecn	ECN-Echo
tcp.flags.urg	Urgent
tcp.flags.ack	Acknowledgment
tcp.flags.push	Push
tcp.flags.reset	Reset
tcp.flags.syn	Syn
tcp.flags.fin	Fin
tcp.flags.str	TCP Flags
tcp.window_size_value	Window
tcp.window_size	Calculated window size
tcp.window_size_scalefactor	Window size scaling factor
tcp.checksum	Checksum
tcp.checksum.status	Checksum Status
tcp.analysis	SEQ/ACK analysis
tcp.analysis.flags	TCP Analysis Flags
tcp.len	TCP Segment Len
tcp.analysis.acks_frame	This is an ACK to the segment in frame
tcp.analysis.bytes_in_flight	Bytes in flight
tcp.analysis.push_bytes_sent	Bytes sent since last PSH flag
tcp.analysis.ack_rtt	The RTT to ACK the segment was
tcp.urgent_pointer	Urgent Pointer
tcp.option_kind	Kind
tcp.option_len	Length
tcp.options	TCP Options
tcp.options.timestamp.tsval	Timestamp value
tcp.options.timestamp.tsecr	Timestamp echo reply
tcp.time_relative	Time since first frame in this TCP stream
tcp.time_delta	Time since previous frame in this TCP stream
tcp.segment_data	TCP segment data
tcp.payload	TCP payload
tcp.analysis.retransmission	This frame is a (suspected) retransmission
tcp.options.nop	TCP Option - No-Operation (NOP)
tcp.options.timestamp	TCP Option - Timestamps

tls	Transport Layer Security
tls.record	Record Layer
tls.record.content_type	Content Type
tls.record.version	Version
tls.record.length	Length
tls.app_data	Encrypted Application Data
udp	User Datagram Protocol
udp.srcport	Source Port
udp.dstport	Destination Port
udp.port	Source or Destination Port
udp.stream	Stream index
udp.length	Length
udp.checksum	Checksum
udp.checksum.status	Checksum Status
udp.time_relative	Time since first frame
udp.time_delta	Time since previous frame
udp.payload	Payload
_ws.expert	Expert Info

Table 2. This table shows most 150 useful features extracted using TShark tool.