Anomaly Detection in Enterprise Networks

B.Tech. Semester Project Report

to be submitted by

Nikhil

B20219

for the partial fulfillment of the degree

of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTER AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

KAMAND-175075, INDIA

November, 2023

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The project aims to address the critical need for real-time detection of various attacks on enterprise network infrastructure distributed over multiple geographical locations over different times. With the increasing sophistication of bad actors, it is crucial to have an equally or more sophisticated system in place that can detect various types of attacks, such as brute force logins, denial of service, malware, and ransomware attacks. Usually one or more types of attacks are launched over multiple nodes of a network over a period of time. In such scenarios, attempting to detect an attack by observing only a single node over a limited time period may either miss detecting the attack or detect it too late. However, by leveraging analysis of logs from network and server endpoints distributed over various geographical locations over a period of time, we can create multi-modal multi-scale spatio-temporal models of such coordinated and correlated attacks for proactive monitoring and early detection to mitigate security risks.

One common approach is to utilize a central server where all subnets share their data to train, build, and update models. However, this method presents various issues. First, transferring data to a centralized cloud would result in significant data transfer overhead, given the size of network data, which accumulates to a substantial amount within a short time frame. Second, sharing data over the internet raises privacy concerns, and data leakage can happen during transmission.

To overcome these challenges, we propose a Federated Learning-based Anomaly Detection System (Fig. 1.1) that enables the distribution of learning to local devices without sending data to a centralized cloud. In federated learning, multiple devices collaborate in training a model while preserving data privacy by keeping the data local. Each local device, or client, has an Anomaly Detection model that can be trained using its own data. The local model is trained for a specified number of epochs and then communicates the model parameters to a master model for aggregation. The shared parameters are aggregated using techniques such as FedAvg [1], and the modified parameters are delivered back to the local clients and they continue training. Thus, collaborative learning can be facilitated without the need to share all data. The local models merely exchange their model parameters, which are significantly smaller in size than the original data.

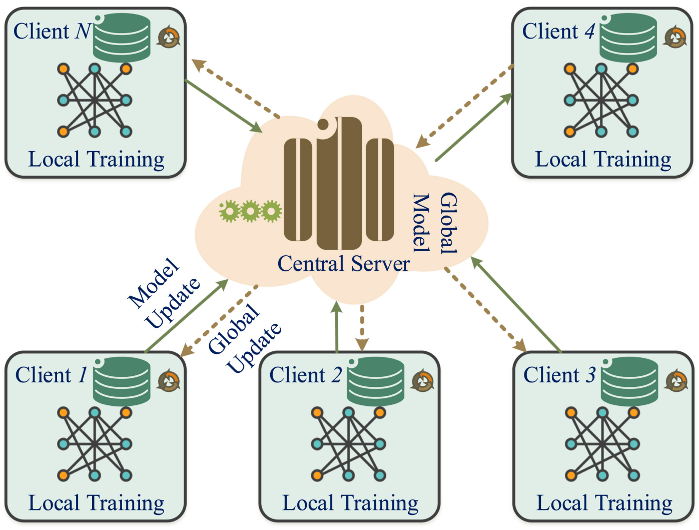


Fig. 1.1: General Federated Learning Architecture

While much progress has been made in Network Anomaly detection and Federated learning, there has been limited application of federated setting in enterprise networks. In the upcoming sections, we will explore related works in this field and their relevance to this work.

1.2 Related Works And their Relevance

S Roy et al. [2] proposes a Federated Learning-based Intrusion Detection System (IDS) for IoT environments, utilizing a Locally Adapted model of federated learning (LAFL). The methodology involves a global model with a stacked autoencoder and feedforward neural network, and multiple local models with personalized layers. The data-volume driven aggregation function is employed for model updates. The study evaluates the method using CICIDS-2017 and NBaIoT datasets, demonstrating the effectiveness of personalized models and data-volume driven aggregation functions in improving the performance of federated learning models for anomaly detection in IoT networks. This differs from ours only in the types of networks covered: it is based on IoT networks while our work involves enterprise networks. Therefore, this proves to be a valuable resource in our task.

J Liu et al. [3] introduces a novel Combined Intrusion Detection System (CIDS) methodology that integrates network and host data to enhance IDS performance. It presents a framework for creating CIDS datasets capable of processing system logs from various operating systems and aligning log entries with network flows. Additionally, it proposes a transformer-based deep learning model named CIDS-Net, designed to incorporate network flow and host-based features of different shapes and dimensions to make better predictions compared to models using only network flow features. The study's methodology offers a thorough approach to integrating network and host-based data for intrusion detection, which we can apply in our application if we also want to utilize host-based data.

L Mohammadpour et al. [4] provides a comprehensive survey of CNN-based network intrusion detection systems (IDSs). It discusses various CNN-based approaches used for detecting network intrusions, anomalies, and attacks. The authors categorize the CNN-IDS approaches and compare their primary capabilities, datasets, architectures, performance metrics, and evaluation methods. Additionally, the paper presents an empirical experiment comparing different approaches based on standard datasets. The survey highlights the increasing use of CNN-based models for intrusion detection and provides valuable insights for researchers in this field.

C He et al. [5] proposes a novel approach called Group Knowledge Transfer (FedGKT) to address the challenge of training large convolutional neural networks (CNNs) in resource-constrained edge devices. FedGKT reformulates Federated Learning (FL) as an alternating minimization approach, which allows for the training of small CNNs on edge devices and the periodic transfer of their knowledge to a large server-side CNN using knowledge distillation. This methodology aims to reduce the computational demand on edge devices, lower communication bandwidth for large CNNs, and enable asynchronous training, while maintaining model accuracy comparable to traditional FL methods.

We will discuss in the next chapter about why CNNs prove to be useful in our work. The last two CNN based approaches discussed provide useful insights on how we can use CNNs in our application. While the first one provides various CNN based approaches for intrusion detection, the second one demonstrates the effectiveness of CNNs in federated setting.

Chapter 2

System Model

Let us briefly describe the goal again:

Goal. *Demonstrate an Anomaly Detection System trained in a Federated Learning environment that achieves comparable performance to a global model trained on all the data, while maintaining privacy and substantially decreasing data transfer overhead.*

In section 2.1, we talk about about the datasets explored for this work, and subsequently in section 2.2, we discuss the methodology adopted.

2.1 Datasets

Table 2.1 presents the datasets examined in this study, number of attacks they cover, as well as some brief comments about them. Please refer to the respective citations for additional details. For our specific use case and to avoid overcomplicating things,

Data-Set Name	**Attack Types**	**Comment**
CIC-IDS2017 [6]	15	Testbed with Firewalls, NAT and all types of OS used
CSE-CIC-IDS2018 on AWS [6]	15	Testbed on AWS environment
NF-UQ-NIDS [7]	20	Fuzzers, Analysis, Backdoor, Exploits etc
SCVIC-CIDS-2022 [8]	15	NIDS and HIDS combined

Table 2.1: Datasets explored in the work

we have considered the CIC-IDS2017 dataset in our work. The following subsection briefly discusses this dataset.

2.1.1 The CIC-IDS2017 Dataset

This dataset is collected over a span of 5 days based on abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols. The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS (Table 2.2). Fig. 2.1 shows the testbed architecture of this dataset. The infrastructure has been divided into two completely separated networks, namely Victim-Network and Attack-Network. In the Victim-Network, they covered all common and necessary equipments, including router, firewall, switch, along with the different versions of the common three operating systems namely Windows, Linux and Macintosh.The Attack-Network includes one router, one switch and four PCs, which have the Kali and Windows 8.1 operating systems. The Victim-Network consists three servers, one firewall, two switches and ten PCs interconnected by a domain controller (DC) and active directory. Also, one port in the main switch of the Victim-Network has been configured as the mirror port and completely captured all send and receive traffic to the network.

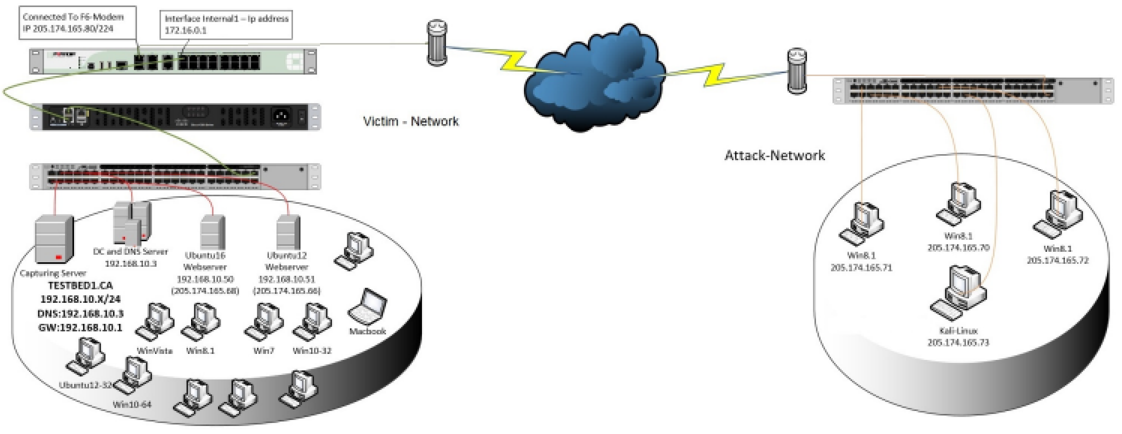


Fig. 2.1: Testbed Architecture of CIC-IDS2017

The dataset is completely labelled and around 80 network traffic features (Table

Attack Type	**Samples**
DoS Hulk	172846
DDoS	128014
DoS GoldenEye	10286
DoS slowloris	5385
DoS Slowhttptest	5228
PortScan	90694
FTP-Patator	5931
SSH-Patator	3219
Bot	1948
Web Attack Brute Force	1470
Web Attack XSS	652
Web Attack Sql Injection	21
Infiltration	36
Heartbleed	11
BENIGN	1592272
Total	2018473

Table 2.2: CIC-IDS2017 Attack Types

2.3 describes those features) extracted and calculated for all benign and intrusive flows by using CICFlowMeter software which is publicly available in Canadian Institute for Cybersecurity website (Habibi Lashkari et al., 2017) [6]. This is a github link for the repository.

2.2 Methodology

2.2.1 Choice of model in federated setting

The model choice involves trade-offs in performance, interpretability, and complexity of implementation of federated setting in that model.

2.2.1.1 Interpretable Models

We previously considered using tree-based models like XGBoost for better interpretability, and found they performed well (comparable to complex deep learning models on this dataset) without federated learning. However, we discovered that while vertical federated settings can be applied to such models [9], applying horizontal federated

Features
Destination Port
Flow Duration
Total Fwd/Bwd Packets
Total Length of Fwd/Bwd Packets
Fwd/Bwd Packet Length Mean,Std,Min,Max
Flow Bytes/s
Flow Packets/s
Flow IAT Mean, Std, Min, Max
Fwd IAT Total, Mean, Std, Min, Max
Bwd IAT Total, Mean, Std, Min, Max
Fwd/Bwd PSH, BRG Flags
Fwd/Bwd Header Length
Fwd/Bwd Packets/s
Packet Length Min, Max, Mean, Std
TCP Flags
Down/Up Ratio
Average Packet Size
Avg Fwd/Bwd Segment Size
Fwd Header Length.1
Fwd/Bwd Avg Bytes/Bulk, Packets/Bulk, Bulk Rate
Subflow Fwd/Bwd Packets, Bytes
Init_Win_bytes_forward, _backward
act_data_pkt_fwd
min_seg_size_forward
Active Time Mean, Std, Max, Min
Idle Time Mean, Std, Max, Min

Table 2.3: CIC-IDS2017 Base Features

settings can be highly inefficient, requiring parameter sharing at each iteration. Additionally, if the feature distributions at each location are very different, such a system would not perform well.

2.2.1.2 FedAvg in Artificial Neural Networks

Due to the reasons stated above, we considered using deep learning models in which federated setting can be easily applied using FedAvg [1]. Fig. 2.2 depicts the basic working of FedAvg. Initially, a global model (an Artificial Neural Network) is initialized. Subsequently, each local model is initialized as a duplicate of the global model, resulting in identical architecture and initialization for each local model. During the training process, in a single iteration, each local model is trained on its respective dataset for a certain number of epochs, after which all local models share their weights with the global model for aggregation. In FedAvg, the aggregation is simply an average (or a weighted average based on the number of samples at each local model). The global model then shares these updated weights with the local models, which in turn update their weights to match the updated weights.

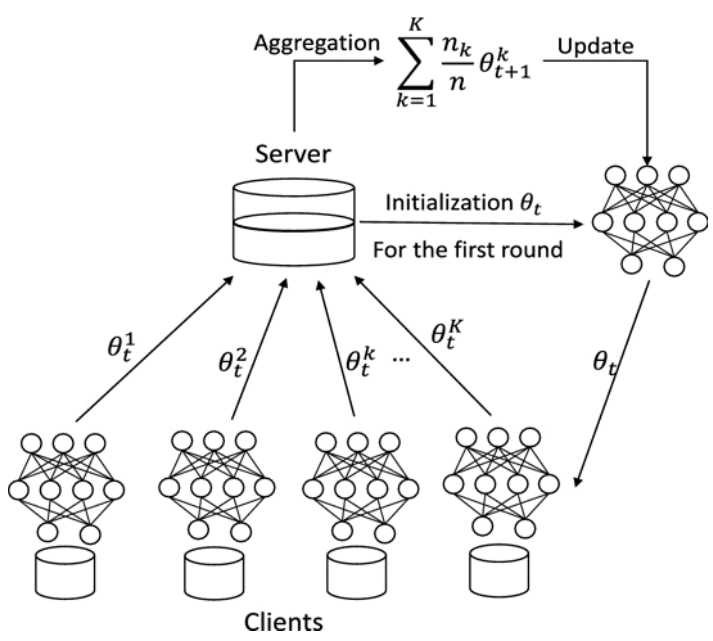


Fig. 2.2: Basic Working of FedAvg

2.2.1.3 CNNs

Specifically, we opted to use the Convolutional Neural Network (CNN) model for this task, for the following reasons. Firstly, the dataset we have already contains features arranged in a way that establishes a spatial relationship, with similar features grouped together. This makes automatic feature extraction using CNN feasible. Secondly, our task does not necessitate local models to specialize in their own locations (the local models can be made to specialize on their own locations by keeping some layers private to the specific location. See LAFL approach in S Roy et al. [2] for more details), as the feature distributions at each location over a significant time period are not significantly different. Thus, we can share all the weights of each local model, simplifying the implementation.

2.2.2 Working

We have implemented the system using Python libraries such as Pandas, Numply, Tensorflow-keras, sklearn and matplotlib. We discuss the implementation steps in the subsequent sections.

2.2.2.1 Preprocessing

Some basic preprocessing steps are done such as removing columns with infinite/NaN values, Z-Score normalization (Standardization), and One Hot Encoding of the Labels. In order to deal with the imbalance and to avoid overfitting on some classes, we have also oversampled some classes, and undersampled the Benign class. We utilized SMOTE for oversampling. Note that the over/under-sampling is only done on the train dataset and the train-test split is already done beforehand. The exact number of samples assigned to each class are shown in Table 2.4.

After that, the train dataset is split into two datasets to simulate the system on two sites. This is discussed in the next section.

Attack Type	Samples
BENIGN	650000
DoS Hulk	200000
DDoS	150000
PortScan	100000
DoS GoldenEye	10000
DoS Slowhttptest	7000
DoS slowloris	7000
FTP-Patator	7000
SSH-Patator	5000
Bot	3000
Web Attack Brute Force	2000
Heartbleed	1500
Infiltration	1500
Web Attack Sql Injection	1500
Web Attack XSS	1500

Table 2.4: Train dataset distribution after oversampling and undersampling

2.2.2.2 Splitting the data to simulate the system on two sites

To demonstrate that the Anomaly Detection System trained in a Federated Learning setting performs on par with the global model trained on the entire dataset, we divided the dataset into two sets representing two sites, each containing very different attack types. This represents the worst-case scenario for any two network locations, where the feature distributions in both networks are highly distinct. Even in this case, we will show that our Anomaly Detection System performs as well as the global model trained on the whole dataset, indicating that the local model at each location can learn about attacks happening on other locations without directly learning from the data of those locations. The idea can be easily extended to n-locations.

Specifically, the split consists of two sites, Site A and Site B. Site A consists 5 attacks which are DOS attacks (DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest and DoS slowloris). Site B consists of rest of the 10 attacks (Bot, FTP-Patator, Heartbleed, Infiltration, PortScan, SSH-Patator, Brute Force Web Attack, Sql Injection Web Attack and XSS Web Attack). Moreover, site A and site B are given 400k and 250k benign samples respectively.

2.2.2.3 Model

As discussed, we are utilizing the CNN model in our work. Fig. 2.3 depicts the overall structure of the model, while Fig. 2.4 presents specifications such as layer size, number of parameters, and so on. The model incorporates 1D convolutions, with convolution layers performing convolutions of size 3. The first pooling layer has a pool size of 4, and the second pooling layer has a pool size of 2. Additionally, dropouts have been included after each layer to facilitate greater generalization, given that the training data has been somewhat altered due to oversampling. Finally, a single dense layer is added. With a total of 157k parameters, this is considered a simple model.

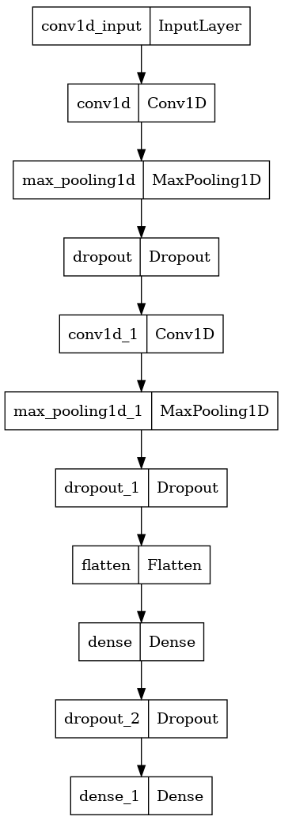
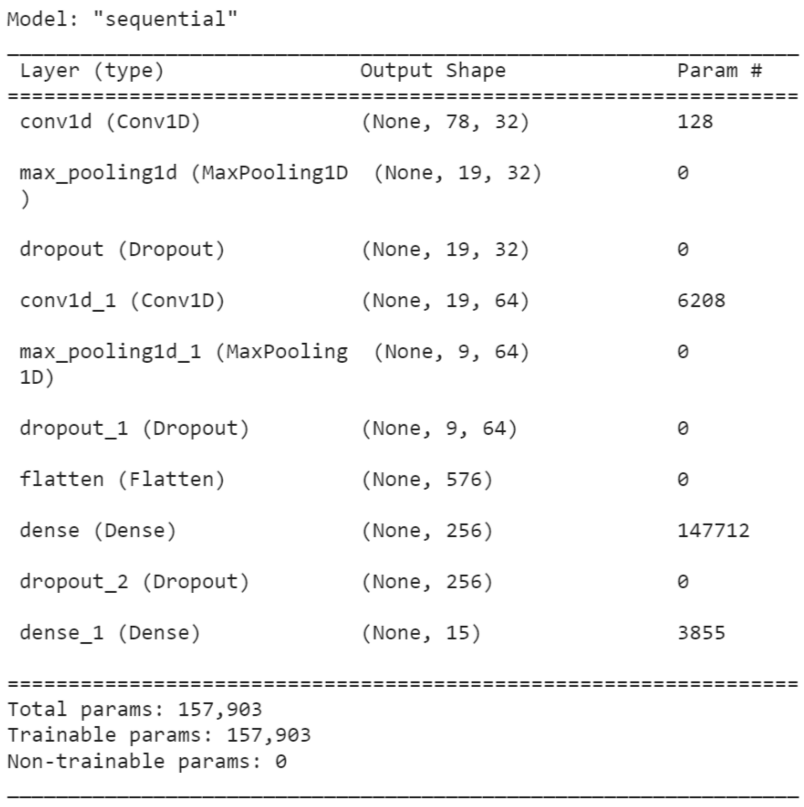


Fig. 2.3: Structure of the CNN model



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 78, 32)            128

 max_pooling1d (MaxPooling1D  (None, 19, 32)           0
 )

 dropout (Dropout)           (None, 19, 32)            0

 conv1d_1 (Conv1D)           (None, 19, 64)            6208

 max_pooling1d_1 (MaxPooling  (None, 9, 64)            0
 1D)

 dropout_1 (Dropout)         (None, 9, 64)             0

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 256)               147712

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 15)                3855

=================================================================
Total params: 157,903
Trainable params: 157,903
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2.4: Specifications of the CNN model

As previously discussed, we use FedAvg to implement federated learning approach for the model. Specifically, during each iteration, we run the local models at their respective sites for 1 epoch before doing the averaging. It takes around 1 minute on each iterations. We achieved accuracy saturation after approximately 20 iterations.

Chapter 3

Results

3.1 Observations

Tables 3.1 and 3.2 depict the model's performance when trained solely on data from site A and site B respectively and tested on test data. Site A's data comprises DoS attacks (the non-orange classes in Table 3.1), while site B consists of the remaining attacks (the non-orange classes in Table 3.2). Consequently, the feature distributions of the training data at these two sites differ significantly. It is clear that these models are unable to predict classes that were not included in their training data, as anticipated.

Table 3.3 illustrates the performance of the model when trained on the combined data of site A and site B, while Table 3.4 illustrates the model's performance in a federated setting. Due to the imbalanced nature of the data, we consider the F1 score to be a better performance metric than accuracy. Both models achieve a similar F1 score, differing by only 0.5%. Therefore, it is evident that the model in the federated setting performs on par with the model trained on combined data.As a result, the federated learning approach proves to be effective in maintaining model performance while preserving data privacy and security. This demonstrates the potential of federated learning in addressing the challenges posed by disparate data distributions across different sites. Furthermore, it highlights the capability of federated learning to enable collaborative model training without the need to centralize sensitive data.

Attack Type	precision	recall	f1-score	support
BENIGN	0.936568	0.980505	0.958033	318547
Bot	0	0	0	390
DDoS	0.996907	0.982072	0.989434	25603
DoS GoldenEye	0.919745	0.980554	0.949176	2057
DoS Hulk	0.860582	0.998872	0.924585	34569
DoS Slowhttptest	0.873396	0.976099	0.921896	1046
DoS slowloris	0.894781	0.987001	0.938631	1077
FTP-Patator	0	0	0	1186
Heartbleed	0	0	0	2
Infiltration	0	0	0	7
PortScan	0	0	0	18139
SSH-Patator	0	0	0	644
Web Attack Brute Force	0	0	0	294
Web Attack Sql Injection	0	0	0	4
Web Attack XSS	0	0	0	130
weighted avg	0.885281	0.931674	**0.907618**	403695

Table 3.1: Results for model at Site A

Attack Type	precision	recall	f1-score	support
BENIGN	0.831368	0.96996	0.895333	318547
Bot	0.382609	0.676923	0.488889	390
DDoS	0	0	0	25603
DoS GoldenEye	0	0	0	2057
DoS Hulk	0	0	0	34569
DoS Slowhttptest	0	0	0	1046
DoS slowloris	0	0	0	1077
FTP-Patator	0.902757	0.994098	0.946228	1186
Heartbleed	0.181818	1	0.307692	2
Infiltration	0.054348	0.714286	0.10101	7
PortScan	0.691861	0.998677	0.817427	18139
SSH-Patator	0.95624	0.916149	0.935765	644
Web Attack Brute Force	0.756757	0.095238	0.169184	294
Web Attack Sql Injection	0.002649	0.5	0.00527	4
Web Attack XSS	0.052676	0.953846	0.099839	130
weighted avg	0.692219	0.815683	**0.74812**	403695

Table 3.2: Results for model at Site B

Attack Type	precision	recall	f1-score	support
BENIGN	0.997648	0.95492	0.975817	318547
Bot	0.591449	0.638462	0.614057	390
DDoS	0.997334	0.978909	0.988035	25603
DoS GoldenEye	0.947546	0.957219	0.952358	2057
DoS Hulk	0.870474	0.999248	0.930426	34569
DoS Slowhttptest	0.873828	0.979924	0.92384	1046
DoS slowloris	0.935626	0.985144	0.959747	1077
FTP-Patator	0.80109	0.991568	0.886209	1186
Heartbleed	0.4	1	0.571429	2
Infiltration	0.151515	0.714286	0.25	7
PortScan	0.737247	0.998346	0.848157	18139
SSH-Patator	0.964052	0.916149	0.93949	644
Web Attack Brute Force	0.804348	0.12585	0.217647	294
Web Attack Sql Injection	0.008696	0.5	0.017094	4
Web Attack XSS	0.055982	0.953846	0.105757	130
weighted avg	0.972801	0.961473	**0.965145**	403695

Table 3.3: Results for global model trained on all data

Attack Type	precision	recall	f1-score	support
BENIGN	0.961011	0.995454	0.977929	318547
Bot	0.945578	0.35641	0.517691	390
DDoS	0.998994	0.969691	0.984124	25603
DoS GoldenEye	0.982564	0.931454	0.956326	2057
DoS Hulk	0.991831	0.944748	0.967717	34569
DoS Slowhttptest	0.877342	0.984704	0.927928	1046
DoS slowloris	0.985323	0.935005	0.959505	1077
FTP-Patator	0.802469	0.986509	0.885023	1186
Heartbleed	0.333333	0.5	0.4	2
Infiltration	0.16129	0.714286	0.263158	7
PortScan	0.988103	0.476211	0.642684	18139
SSH-Patator	0.965517	0.913043	0.938547	644
Web Attack Brute Force	0	0	0	294
Web Attack Sql Injection	0	0	0	4
Web Attack XSS	0.012048	0.069231	0.020525	130
weighted avg	0.965729	0.963817	**0.96027**	403695

Table 3.4: Results for model in federated setting

3.2 Conclusion

- The proposed Federated Learning model with CNN is effective for enterprise networks with multiple sites and delivers a similar performance as a single model trained on combined data.
- We have successfully tested our model by training it on two sites, with the capability to easily extend to multiple sites. These sites have very different feature distributions, but in our domain, the feature distributions across different sites within the same enterprise are generally similar. Therefore, we anticipate that our system will perform even better in the original situation.
- The proposed model eliminates the need to send data to a central location and addresses privacy and data transfer requirements.
- The proposed model, with only 157k parameters and a size of less than 1MB, can be sent across the network using a secure channel with very little overhead.
- Our objective was to showcase an Anomaly Detection System trained in a Federated Learning environment that achieves similar performance to a global model trained on all the data, while preserving privacy and significantly reducing data transfer overhead. It is evident that we have successfully achieved this goal.

Chapter 4

Future Work

4.1 Timeline and Progress

Fig. 4.1 depicts the scheduled timeline for the project. It aligns with the original timeline we established for the work. We have currently met the targeted objective for November end, 2023, which was the implementation of a 2 node Federated Learning mode for offline detection. We are optimistic about achieving the upcoming objectives outlined in the gantt-chart.

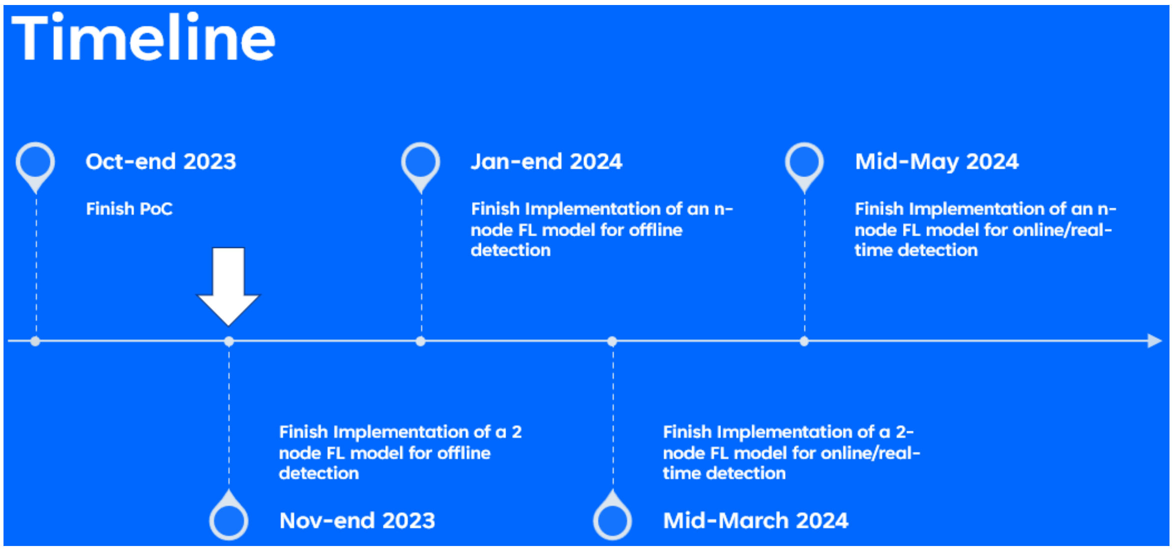

Timeline
Oct-end 2023
Finish PoC
Jan-end 2024
Finish Implementation of an n-node FL model for offline detection
Mid-May 2024
Finish Implementation of an n-node FL model for online/real-time detection
Finish Implementation of a 2 node FL model for offline detection
Nov-end 2023
Finish Implementation of a 2-node FL model for online/real-time detection
Mid-March 2024

Fig. 4.1: Timeline of the project

Our goal is to extend the current work by allowing the system to be able to work on any arbitrary number of sites and to be able to have online training and real time detection, which can be deployed within enterprise networks.

4.2 Scope for improvement

There are some things in this work which could be improved. We list them below:

- We are currently utilizing the CIC-IDS2017 dataset, which is widely recognized as a benchmark dataset. It serves as a strong validation of the system's performance with real-time data. An additional step to further strengthen this validation would be to test the system on additional datasets.
- At present, we are utilizing the FedAvg federated learning algorithm, which is a relatively straightforward method that relies on averaging. The weights are determined based on the number of samples at each site. To enhance the model's performance and tailor it to this domain, we plan to explore parameter estimation-based methods that can more accurately weight the contributions of each site. This will allow us to optimize the model for improved performance.
- At present, our classification process involves determining whether each packet contains an anomaly or not. However, to better address certain attacks that exhibit temporal patterns, we are considering the implementation of sequential models such as RNNs or Transformers. These models will enable us to capture and analyze the temporal aspects of the data, enhancing our ability to detect and respond to specific attack patterns.
- The current work only processes Network based data for anomaly detection. However, it can be beneficial to also consider Host based data for anomaly detection as described in J Liu et al. [3].
- In order to handle the imbalanced nature of the dataset, we are currently relying on SMOTE based oversampling to some extent. However, we can employ better

methods such as using Generative Adversarial Networks (GANs) [10] to generate more realistic data in order to balance the data.

- Additionally, we can integrate rule-based and threshold-based models in the initial layers of the pipeline before feeding the data into the main model. This preemptive filtering can effectively identify and discard obvious suspicious packets, allowing the main model to focus on training to detect more complex and elusive attacks, and not overtrain on the easily detectable anomalies.
- As we had shown in the gantt-chart (Fig. 4.1), we plan to employ online training in our system. This would require us to capture real time data and also to process it to be used in our system, which would require us to construct a pipleline for the same.
- Additionally, adding a dashboard and reporting mechanism for suspicious packets would be beneficial to the system.

Bibliography

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

[2] S. Roy, J. Li, and Y. Bai, "Federated learning-based intrusion detection system for iot environments with locally adapted model," in *2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2023, pp. 203–209.

[3] J. Liu, M. Simsek, B. Kantarci, M. Bagheri, and P. Djukic, "Collaborative feature maps of networks and hosts for ai-driven intrusion detection," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 2662–2667.

[4] L. Mohammadpour, T. C. Ling, C. S. Liew, and A. Aryanfar, "A survey of cnn-based network intrusion detection," *Applied Sciences*, vol. 12, no. 16, p. 8162, 2022.

[5] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 068–14 080, 2020.

[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[7] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," in *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10*. Springer, 2021, pp. 117–135.

[8] J. Liu, M. Simsek, B. Kantarci, M. Bagheri, and P. Djukic, "Scvic-cids-2022: Bridging networks and hosts via machine learning-based intrusion detection," 2022. [Online]. Available: https://dx.doi.org/10.21227/dn9v-3278

[9] Q. M. Nguyen, N. K. Le, and L. M. Nguyen, “Scalable and secure federated xgboost,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[10] M. H. Shahriar, N. I. Haque, M. A. Rahman, and M. Alonso, “G-ids: Generative adversarial networks assisted intrusion detection system,” in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2020, pp. 376–385.