SDN-based Federated Learning System for Detecting Infected IoT Devices

Han Yang, Nathanael Bowley, Hongwei Zhang, Raham Moghaddam, Ehssan Mousavipour

DALHOUSIE UNIVERSITY

April 13, 2023

Outline

1. Introduction
2. Background and Related Works
3. System Design
4. Implementation
5. Evaluation
6. Conclusion and Future Directions
7. Demonstration

Introduction

- IoT devices are becoming ubiquitous.
- Internet-accessible IoT devices predicted to reach 29.42 billion by 2030.
- Mirai malware created in September 2016
  - IoT devices exploited in botnets, DoS, and DDoS attacks
  - Detection and prevention methods crucial for IoT security
- Proposed project builds on existing research from Nguyen et al., and Popoola et al.
- Train and deploy federated learning-based approach to detect infected IoT devices by malware.
- Aim to emulate a Small Office/Home Office (SOHO) IoT environment with SDN approach.

Internet of Things (IoT)

- Network of interconnected devices
- Embedded with sensors, software, and connectivity, which enables these objects to collect and exchange data

Figure 1. IoT architecture [4]

Malware (Mirai)

- It turns networked devices running into remotely controlled bots
- Use these compromised devices to launch attack such as DDoS on the victims.

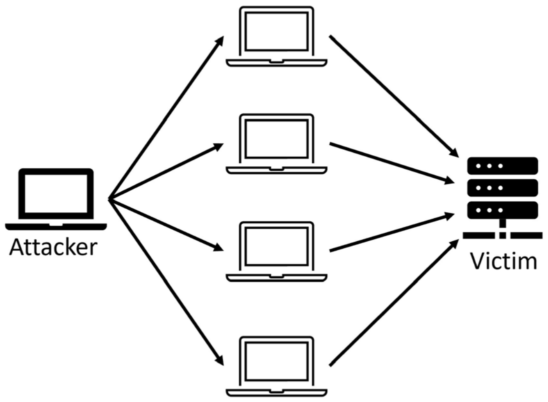


Figure 2. DDoS generated by IoT botnets

Software-Defined Networking (SDN)

- Separation of control and data plane
- Flow based packet forwarding
- Network Operating Systems (NOS) is the controller with a global network view
- Programmable network, i.e., applications run on the NOS

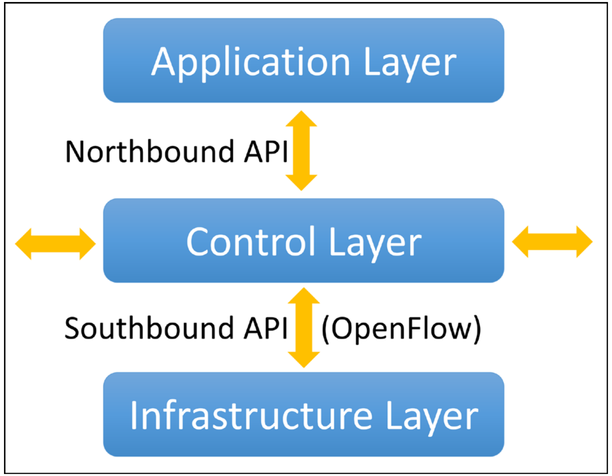


Figure 3. SDN architecture

Federated Learning (FL)

- It is a machine learning approach that enables multiple devices to collaboratively train a shared model while keeping their data locally.
- Locally trained model feed into the Global model

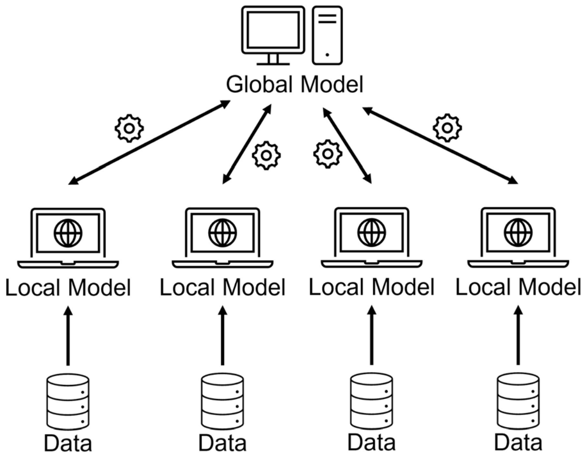


Figure 4. Federated Learning

Motivation

- Given many heterogeneous IoT devices can be connected to an organizational network, we want to deploy a highly effective and accurate method in detecting compromised IoT devices which have been infected and act to mitigate.
- Federated Learning shown its high classification performance even in a data imbalance and massively distribution environment [1].
- In SDN, once the malicious devices have been found, stakeholders can take response to block infected device quick by setting firewall rules.

Related Work

- ML-based Intrusion Detection Systems (IDS) for IoT devices are efficient due to their accuracy and flexibility
- Traditional ML models require large amounts of data, but Federated Learning (FL) uses distributed ML and needs less data
- FL allows clients to contribute to ML models without sharing full data, providing a solution for ML-based IDS in IoT attacks
- Previous works on malware detection used FL for IoT devices, inspiring the current project
- Popoola et al. proposed a zero-day botnet detection method for IoT-edge devices based on Federated Deep Learning (FDL)

Related Work

- FDL-based methods achieved better classification results with more security channels during model aggregations.
- Nguyen et al. designed an anomaly IDS based on FL called DIoT, achieving a zero false alarm rate and 95.6% average true positive rate.
- Both previous works deployed the ML-based IDS on a traditional network for IoT malware detection.
- Maeda et al. proposed a new IoT botnet detection system based on SDN architecture.
- SDN allows for faster detection and response to attacks, reducing damage by isolating infected hosts using VLANS.

Related Work

Figure 4 displays previous FL-based attack detection systems. Most previous FL-based IDS focused on malware detection without focus on DDoS detection for IoT

Our claimed contributions to build upon the literature:
1) Our project will be the first FL-based approach to detect infected IoT devices which is deployed under SDN architecture.
2) Our project focuses on DDoS detection on botnet infected IoT devices which has not been covered by previous research
3) Our system will set up open flow rules to block DDoS traffic of Mirai botnet infected devices when attacks are detected to reduce the impact of the attacks

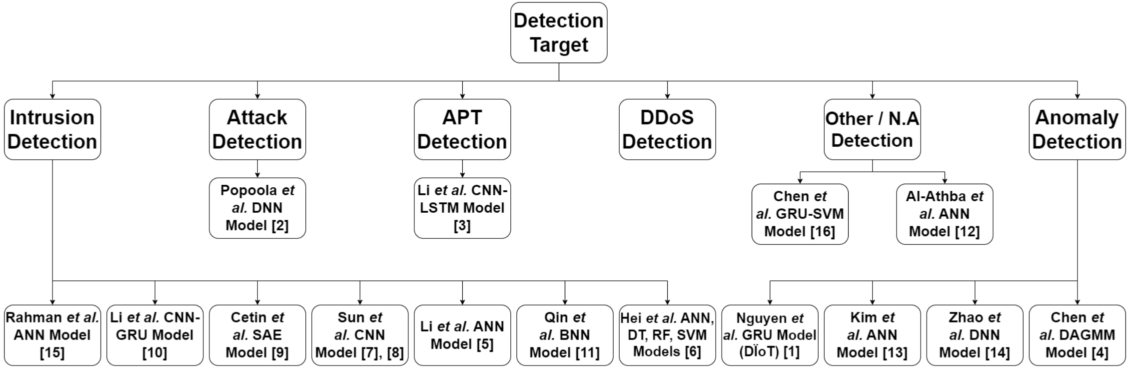


Figure 5. Related works

System Design

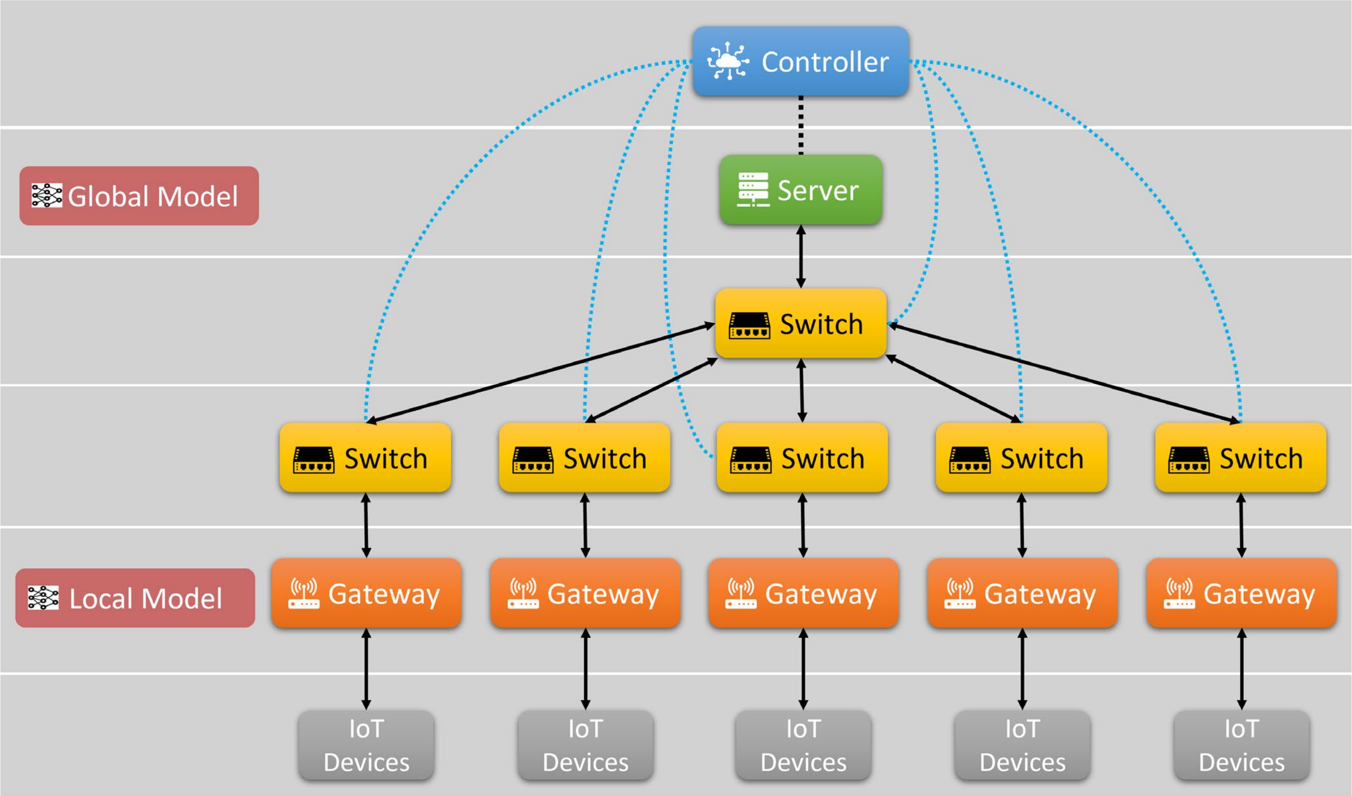


Figure 6. System architecture

Implementation

- Data preprocessing
  - Separated the datasets to different portions based on the devices
  - Clean NAN, perform normalization

- Architecture of neural network
  - 115 neurons in input layers
  - 4 hidden layers each have 100 neurons

```
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=115, out_features=100, bias=True)
    (1): ReLU()
    (2): Linear(in_features=100, out_features=100, bias=True)
    (3): ReLU()
    (4): Linear(in_features=100, out_features=100, bias=True)
    (5): ReLU()
    (6): Linear(in_features=100, out_features=100, bias=True)
    (7): ReLU()
    (8): Linear(in_features=100, out_features=100, bias=True)
    (9): ReLU()
    (10): Linear(in_features=100, out_features=5, bias=True)
    (11): Softmax(dim=1)
  )
)
```

Implementation

- Implementation of the federated learning
  - Implemented based on python library called flower
- Mininet setup
  - Controller to take the responsibility for the network management.
  - Hosts act as security gateway and server, used xterm to interact with them.

```python
class FlowerClient(fl.client.NumPyClient):
    def __init__(self, net, trainloader, valloader, loss_func, optimizer, epoch):
        self.net = net
        self.trainloader = trainloader
        self.valloader = valloader
        self.loss_func = loss_func
        self.optimizer = optimizer
        self.epoch = epoch

    def get_parameters(self, config):
        return get_parameters(self.net)

    def fit(self, parameters, config):
        set_parameters(self.net, parameters)
        train(self.trainloader, self.net, self.loss_func, self.optimizer, self.epoch)
        return get_parameters(self.net), len(self.trainloader), {}
```

```python
fl.client.start_numpy_client(
    server_address = "10.0.0.1:8080",
    client=client1,
)
```

Dataset

N-BaIoT Dataset to Detect IoT Botnet Attacks [3]

- The dataset include real traffic data collected from 9 commercial IoT devices that were infected by Mirai and BASHLITE
- We selected data for five Mirai-infected IoT devices
    - •Benign
    - •ACK: acknowledgement flooding
    - •SCAN: automatic scanning for vulnerable devices
    - •SYN: synchronization flooding
    - •UDP: UDP flooding

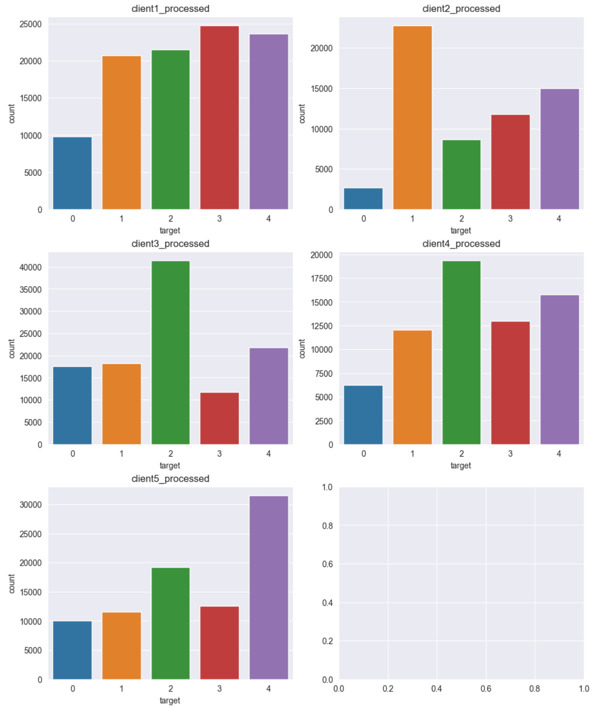


Figure 7. Original dataset distribution

Dataset

Dataset balancing methods:

- Under-sampling: random under-sampling
- Over-sampling: Synthetic Minority Oversampling TEchnique (SMOTE)

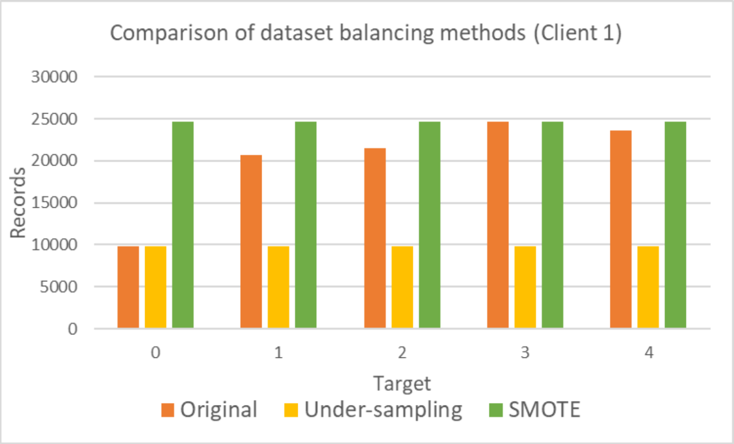


Figure 8. Client 1 Dataset comparison

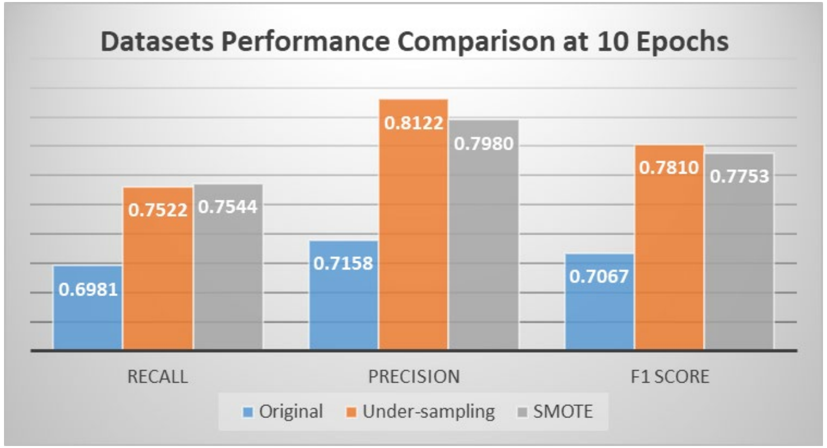


Figure 9. Dataset comparison

Evaluation

- Utilized NVME (Non-Volatile Memory Express, which is a protocol designed specifically for accessing high-speed storage devices like SSDs) SSD to store Kali Linux along with Ryu and Mininet
- Encountered issues with Ryu, which prevented us from using Kali Linux and forced us to use a VM
- Due to limitations of VM's current version, we were unable to Leverage GPU acceleration, and thus, had to use CPU

TABLE I
TESTING PLATFORM

Processor	Intel Core i9 13900K, 5.55 GHz
Video Card	NVIDIA GeForce RTX 3070 Ti
Motherboard	ASUS ROG Strix Z790-E Gaming WiFi
Memory	32 GB, LPDDR5, 4800 MHz
Hard Drive	2 TB, M.2, PCIe NVMe, SSD
Operating System	Windows 11 Home
Virtual Machine	Oracle VM VirtualBox Operating System: Ubuntu (64-bit) Base Memory: 23865 MB Processors: 4 Storage: 64 GB

Figure 10. Testing Machine

Evaluation

- To train our Federated Learning model, we utilized a custom implementation of the Federated Averaging algorithm
- Our training process involved 10 rounds, with testing conducted at 5, 10, 15, 20, 30, 40, and 50 epochs to monitor the model's performance
- We collected weighted averages of the following metrics at each round to generate graphs for clients:
  - Precision
  - Recall
  - F1 Score

$$TP = \text{True Positives}$$
$$FP = \text{False Positives}$$
$$TN = \text{True Negatives}$$
$$FN = \text{False Negatives}$$

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

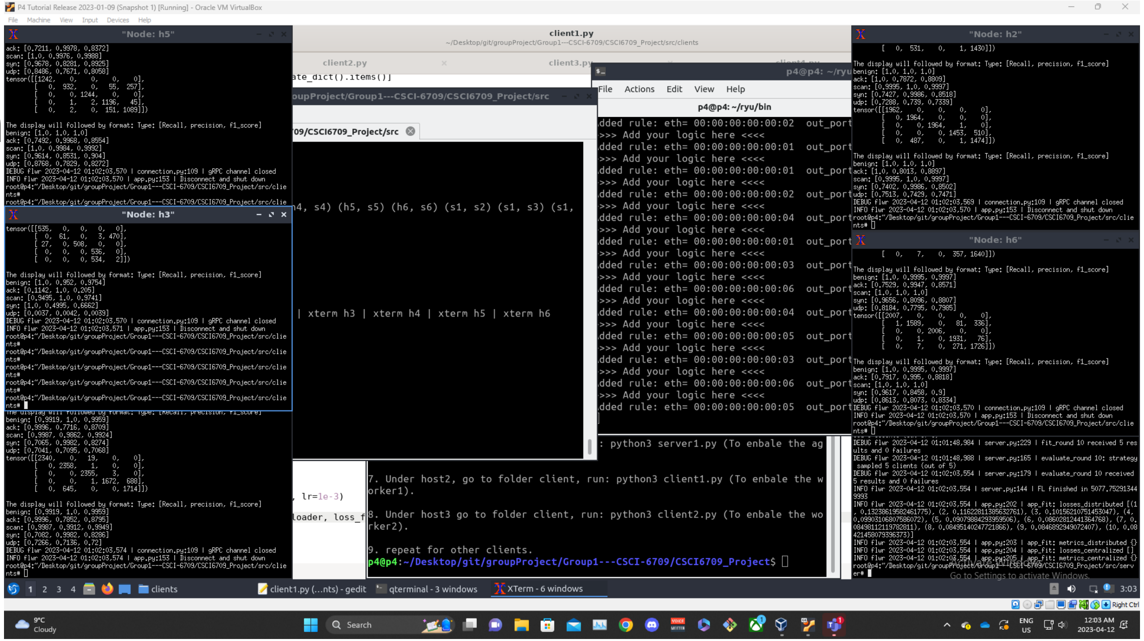
$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \quad (4)$$

Evaluation

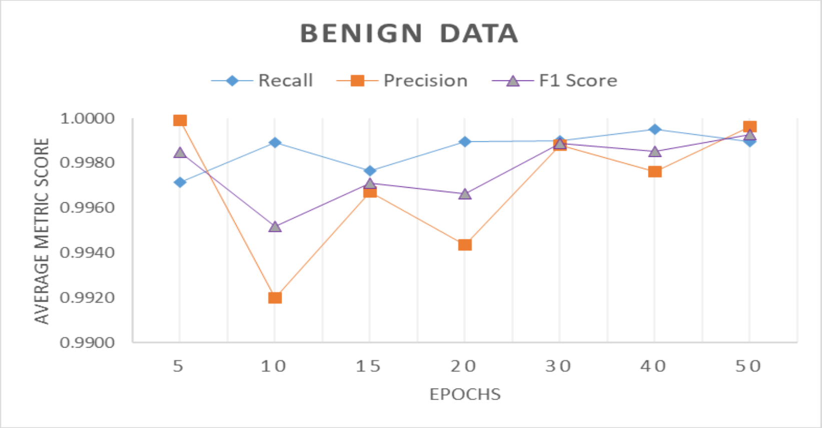
- Data generated from 5 clients during testing was aggregated into an average matrix of labels by metrics
- This average matrix was used to evaluate the impact of modifying the epochs on the performance of our model

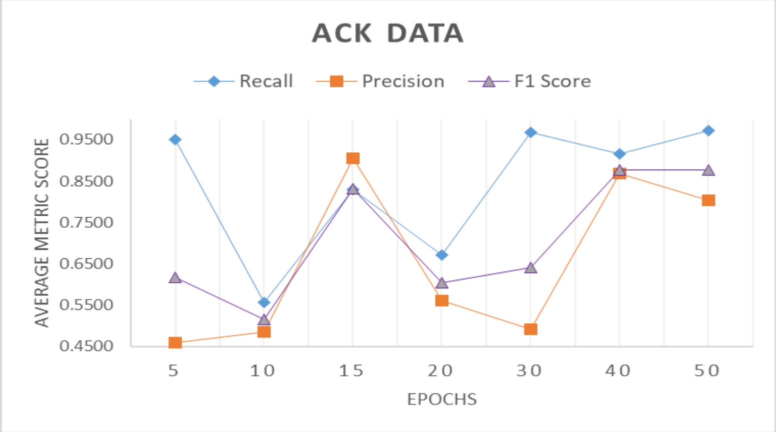


Evaluation

Benign			
Epoch	Recall	Precision	F1 Score
5	0.997160	0.999900	0.998500
10	0.998920	0.992000	0.995180
15	0.997644	0.996699	0.997105
20	0.998980	0.994360	0.996640
30	0.999000	0.998800	0.998900
40	0.999506	0.997611	0.998515
50	0.998980	0.999620	0.999280

ACK			
Epoch	Recall	Precision	F1 Score
5	0.951520	0.458780	0.618060
10	0.556740	0.485540	0.516380
15	0.828962	0.905049	0.831146
20	0.671825	0.561650	0.605375
30	0.969200	0.491500	0.641400
40	0.915589	0.868025	0.877950
50	0.973440	0.804200	0.877580

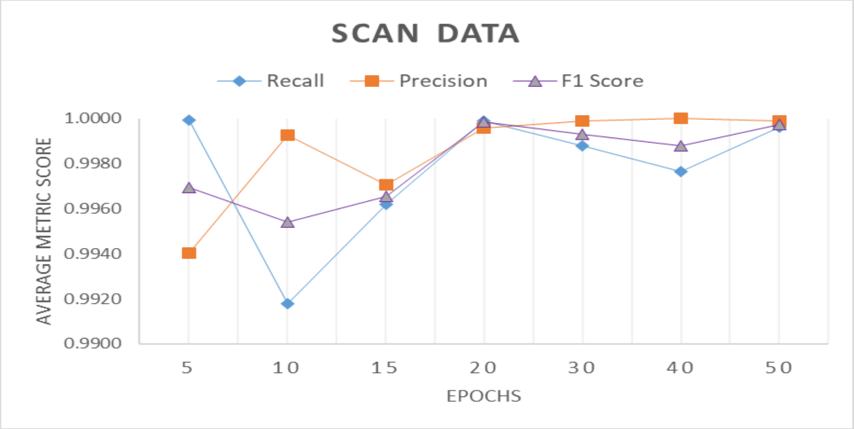


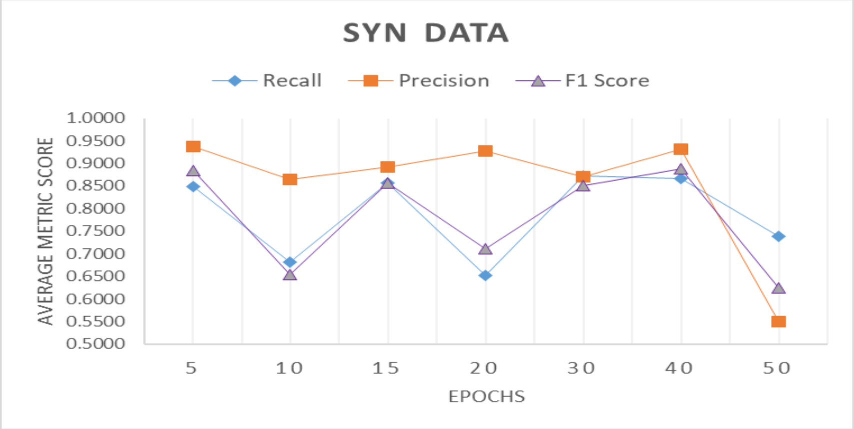


Evaluation

SCAN			
Epoch	Recall	Precision	F1 Score
5	0.999920	0.994040	0.996940
10	0.991780	0.999260	0.995420
15	0.996202	0.997053	0.996562
20	0.999900	0.999575	0.999875
30	0.998800	0.999900	0.999300
40	0.997659	1.000000	0.998810
50	0.999620	0.999900	0.999740

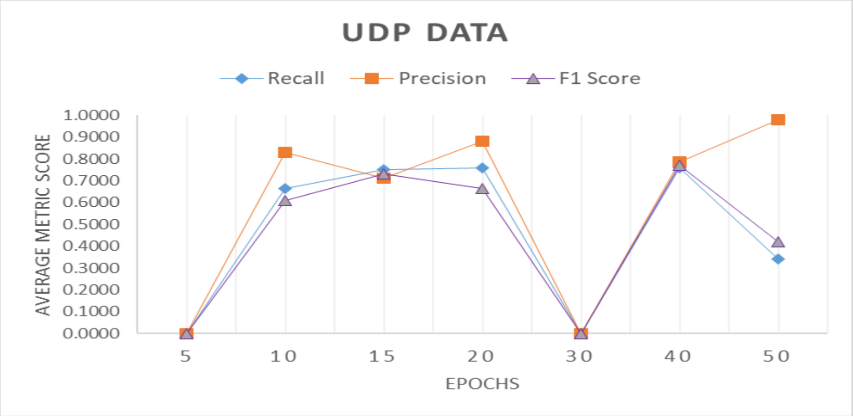
SYN			
Epoch	Recall	Precision	F1 Score
5	0.849340	0.937216	0.883400
10	0.681100	0.864500	0.653780
15	0.856472	0.892395	0.857089
20	0.652425	0.928325	0.711475
30	0.872400	0.870900	0.851700
40	0.867229	0.931396	0.888697
50	0.738120	0.549900	0.624180





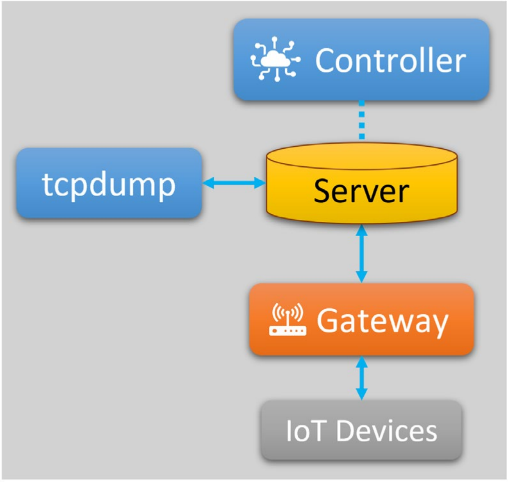
Evaluation

UDP			
Epoch	Recall	Precision	F1 Score
5	0.000000	0.000000	0.000000
10	0.664080	0.829420	0.609400
15	0.752239	0.711100	0.730773
20	0.756725	0.881850	0.662750
30	0.000000	0.000000	0.000000
40	0.758857	0.785450	0.770799
50	0.340220	0.978140	0.418860



Evaluation

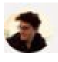
- We also have evaluated the communication overhead for the federated learning communication traffic
- We runed a packet sniffer on the server, captured all traffic then filtered by dst.port 8080 to find the FL communication traffics.
- Then we converted the traffics to csv then calculated the communications overhead by summing up their lengths
    - Found the communications overhead for FL is **3057833** bytes total



ip.src	ip.dst	eth.src	eth.dst	frame.len	frame.tim	relative_time
10.0.0.2	10.0.0.1	12:e4:98:b	c6:1b:49:d	74	1.68E+09	0
10.0.0.2	10.0.0.1	12:e4:98:b	c6:1b:49:d	66	1.68E+09	0.002306
10.0.0.2	10.0.0.1	12:e4:98:b	c6:1b:49:d	66	1.68E+09	0.004528
10.0.0.2	10.0.0.1	12:e4:98:b	c6:1b:49:d	383	1.68E+09	0.025098

Future Directions

- We used a simple Deep Neural Network to detect the attacks and benign traffic.
- In future work, we will add Convolutional Neural Network (CNN) to compare the performances
- CNNs use convolutional layers to extract features from input data, followed by pooling layers to reduce the spatial dimensions of the output, and then fully connected layers to classify the traffics based on the extracted features
- Based on the previous works in the field, we believe we achieve a better performance using CNN
- The future direction for our final paper is to also test by varying the number of rounds to see if rounds have a statistically significant impact on our model's accuracy.
- We also want to test by varying the number of clients to see the impact on the model.
- We also want to test with real time traffic data using tools such as tcpReplay in future work.



Conclusion

- A simple neural network was used to classify Benign, ACK, SYN, SCAN, and UDP traffics.
- The model performed very well at detecting Benign traffic, and very well at detecting SCAN traffic.
- The model performed slightly worse in detecting ACK traffics, as the precision was not high.
- The model performed poorly on both UDP an SYN attacks.
- Overall, results indicate that our Neural Network was not able to find out all the different patterns in our dataset, possibly due to a simple architecture. Likewise preprocessing or distribution may have played a factor.

Reference

1. T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan and A. -R. Sadeghi, "DÏoT: A Federated Self-learning Anomaly Detection System for IoT," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, 2019, pp. 756-767, doi: 10.1109/ICDCS.2019.00080.
2. S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh and O. Jogunola, "Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT-Edge Devices," in *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930-3944, 1 March1, 2022, doi: 10.1109/JIOT.2021.3100755.
3. Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici 'N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders', *IEEE Pervasive Computing*, Special Issue - Securing the IoT (July/Sep 2018).
4. Yousefnezhad, Narges & Malhi, Avleen & Keyriläinen, Tuomas & Främling, Kary. (2023). A Comprehensive Security Architecture for Information Management throughout the Lifecycle of IoT Products. *Sensors*. 23. 3236. 10.3390/s23063236.

5. 1."Financial Impact of Mirai DDoS Attack on Dyn Revealed in New Data." Corero, 03-Aug-2017. [Online]. Available: https://www.corero.com/financial-impact-of-mirai-ddos-attack-on-dyn-revealed-in-new-data/. [Accessed: 15-Mar-2023].
6. 2.P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” in Proceedings. Presses universitaires de Louvain, 2015, p. 89.
7. 3.A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using Recurrent Neural Networks (RNN)," 2016 Integrated Communications Navigation and Surveillance (ICNS), Herndon, VA, USA, 2016, pp. 5C2-1-5C2-8, doi: 10.1109/ICNSURV.2016.7486356.
8. 4.M. Du, F. Li, G. Zheng, and V. Srikumar, “Deeplog: Anomaly detection and diagnosis from system logs through deep learning,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS ’17. ACM, 2017, pp. 1285–1298, http://doi.acm.org/10.1145/3133956.3134015.
9. 5.A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, “Detection of early-stage enterprise infection by mining large-scale log data,” in Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on. IEEE, 2015, pp. 45–56.
10. 6.C. Krugel, T. Toth, and E. Kirda, “Service specific anomaly detection ¨ for network intrusion detection,” in Proceedings of the 2002 ACM symposium on Applied computing. ACM, 2002, pp. 201–208.
11. 7.L. Portnoy, E. Eskin, and S. Stolfo, “Intrusion detection with unlabeled data using clustering,” in Proceedings of ACM CSS Workshop on Data Mining Applied to Security, 2001.
12. 8.S. Rajasegarar, C. Leckie, and M. Palaniswami, “Hyperspherical cluster based distributed anomaly detection in wireless sensor networks,” Journal of Parallel and Distributed Computing, vol. 74, no. 1, pp. 1833– 1847, 2014.
13. 9.A. Kleinmann and A. Wool, “Automatic construction of statechartbased anomaly detection models for multi-threaded scada via spectral analysis,” in Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy, ser. CPS-SPC ’16. ACM, 2016, pp. 1–12.
14. 10.R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, “Specification-based anomaly detection: a new approach for detecting network intrusions,” in Proceedings of the 9th ACM conference on Computer and communications security. ACM, 2002, pp. 265–274.
15. 11.Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao, and A. Prakash, “ContexloT: Towards providing contextual integrity to appified IoT platforms,” in 24th Annual Network & Distributed System Security Symposium (NDSS), feb 2017.
16. 12.S. Raza, L. Wallgren, and T. Voigt, “Svelte: Real-time intrusion detection in the internet of things,” Ad hoc networks, vol. 11, no. 8, pp. 2661– 2674, 2013.
17. 13.R. Doshi, N. Apthorpe, and N. Feamster, “Machine learning ddos detection for consumer internet of things devices,” CoRR, vol. abs/1804.04159, 2018. [Online]. Available: http://arxiv.org/abs/1804.04159
18. 14.C. Krugel, T. Toth, and E. Kirda, “Service specific anomaly detection ¨ for network intrusion detection,” in Proceedings of the 2002 ACM symposium on Applied computing. ACM, 2002, pp. 201–208.
19. 15.L. Portnoy, E. Eskin, and S. Stolfo, “Intrusion detection with unlabeled data using clustering,” in In Proceedings of ACM CSS Workshop on Data Mining Applied to Security, 2001.
20. 16.S. Rajasegarar, C. Leckie, and M. Palaniswami, “Hyperspherical cluster based distributed anomaly detection in wireless sensor networks,” Journal of Paralle
21. 17.M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, “Understanding the mirai botnet,” in 26th USENIX Security Symposium (USENIX Security 17). Vancouver, BC: USENIX Association, 2017, pp. 1093–1110.
22. 18.S. Edwards and I. Profetis, “Hajime: Analysis of a decentralized internet worm for IoT devices,” Rapidity Networks, Tech. Rep., 2016.
23. 19.C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, “DDoS in the IoT: Mirai and other botnets,” Computer, vol. 50, no. 7, pp. 80–84, 2017.
24. 20.Radware, “BrickerBot results in PDoS attack,” https://security.radware.com/ddos-threats-attacks/brickerbot-pdospermanent-denial-of-service/, 2017, [Online; accessed 8-April-2019]

Demonstration

Exit the slide and present the program

Thanks for watching.
Any questions?