Adversarial Trained Deep Learning Poisoning Defense: SpaceTime

Christian Dunham

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2022

Committee:
Geetha Thamilarasu
Brent Lagesse
Afra Mashhadi

Program Authorized to Offer Degree:
Computing and Software Systems

University of Washington

Abstract

Adversarial Trained Deep Learning Poisoning Defense:
SpaceTime

Christian Dunham

Chair of the Supervisory Committee:
Geetha Thamilarasu
Department of Computing and Software Systems

Smart homes, hospitals, and industrial complexes are increasingly reliant on the Internet of Things (IoT) technology to unlock doors, regulate insulin pumps, or operate critical national infrastructure. While these technologies have made tremendous improvements that were not achievable before IoT, the increased the adoption of IoT has also expanded the attack surface and increased the security risks in these landscapes. Diverse IoT protocols and networks have proliferated allowing these tiny sensors with limited resources to both create new edge networks and deploy at depth into conventional internet stacks. The diverse nature of the IoT devices and their networks has disrupted traditional security solutions.

Intrusion Detection Systems (IDS) are one security mechanism that must adopt a new paradigm to provide measurable security in this technological evolution. The diverse resource limitations of IoT devices and their enhanced need for data privacy complicates centralized machine learning models used by modern IDS for IoT environments. Federated Learning (FL) has drawn recent interest adapting solutions to meet the requirements of the unevenly distributed nodes in IoT environments. A federated anomaly-based IDS for IoT adapts to the computational restraints, privacy needs, and heterogeneous nature of IoT networks.

However, many recent studies have demonstrated that federated models are vulnerable to poisoning attacks. The goal of this research is to harden the security of federated learning

models in IoT environments to poisoning attacks. To the best of our knowledge poisoning defenses do not exist for IoT. Existing solutions to defend against poisoning attacks in other domains commonly utilize different spatial similarity measurements from Euclidean Distance (ED), cosine similarity (CS), and other pairwise measurements to identify poison attacks.

Poisoning attack methodologies have also adapated to IoT causing an evolution that defeats these existing defensive solutions. Poisoning evolution creates a need to develop new defensive methodologies. In this we develop *ST-DSD* a deep learning recurrent neural network that uses a four dimensional spacetime manifold to distinguish federated participants. *ST-DSD* is built upon a time series regression many-to-one architecture to provide an adversarial trained defense for federated learning models. Simulation results shows that *ST-DSD* exceeds the previous solutions for Byzantine and Sybil label flipping, backdoor, and distributed backdoor attacks in an IoT environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ATTACK RATE: The ratio of attackers to honest clients in an experiment.

BACKDOOR ATTACK: A poisoning attack where the attacker constructs a 'backdoor' to the models output by attacking a set of parameters (trigger) that will control the prediction of the model for that trigger while not affecting the models convergence of other parameters.

BI-DIRECTIONAL LONG-SHORT TERM MEMORY (BI-LSTM): An LSTM that that trains on the input sequence with two LSTMs. The first LSTM trains on the sequence as-is, the second trains on a reversed copy of the input sequence.

CONVERGENCE: The values of a process that have a tendency in behavior over time. Often time measured through model loss or accuracy.

DENSE LAYER: A neural network layer where each neuron receives input from all the neurons in the previous layer. Dense layers perform matrix-vector multiplication. Dense layers have a fixed vector output.

DISTRIBUTED BACKDOOR ATTACK: A backdoor attack that further breaks the trigger of the backdoor attack down to several separate 'sub-triggers' to avoid detection while still achieving creating a backdoor to the models prediction.

FEDERATED LEARNING (FL): Machine learning method that enables machine learning from different data sets originating from distributed locations without sharing data.

INTERNET OF THINGS (IOT): Networking capability that allows information to be sent and received from sensors and objects such as smart lights, and medical devices.

INTRUSION DETECTION SYSTEM (IDS): An application that monitors network traffic and logs threats and or suspicious activity.

LABEL FLIP ATTACK: A poisoning attack in which the attacker changes the class labels of their portion of the data set.

LONG-SHORT TERM MEMORY (LSTM): Cells or blocks of an RNN that utilize a series of gates to simulate human thought. LSTMs are capable of learning which information to hold and carry forward (long term) as well as the typical RNN feedfoward short memory. This also allows LSTMs to decide which information is lacks the importance to remember and carry forward.

POISONING ATTACK: An adversarial method to corrupt the data used in machine learning with the end goal of manipulating the models convergence.

POISON RATE: The ratio of accepted poison weights to accepted honest client weights by the machine.

RECURRENT NEURAL NETWORK: An artificial neural network that can process sequential data, recognize patterns, and predict a final output. RNNs are recurrent and different from other neural networks in that they can repeatedly perform the same task on a sequence of inputs. RNNs have been historically used to solve speech recognition and natural language problems.

TIMEDISTRIBUTED DENSE LAYER: A neural network layer that applies the computations of that layer to each output of the sequence.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to University of Washington at Bothell, where he has had the opportunity to work with Dr. Geetha Thamilarasu, Dr. Lagesse, and Dr. Mashaddi. This journey would not have been possible without Suzanna Martinez for her constant counsel and advise. The following faculty instilled a passion for problem solving: Dr. Stiber, Dr. Dimpsey, Dr. Dupuis, and Dr. Si.

De Oppresso Liber

DEDICATION

to my family: Flora, Arturo, and Sofia – for your support and sacrifice.

Chapter 1

INTRODUCTION

The Internet of Things (IoT) ecosystem has rapidly expanded into mainstream facets of our daily lives, industrial process, and critical national infrastructure resulting in much research in current years. RFID and IPv6 birthed a heterogenous device race that Kevin Ashton minted as the 'Internet of Things to describe the ubiquitous sensors in 1999 [17]. Since then, IoT devices have been incorporated into applications from smartwatches to implanted medical actuators. The sensors utilize Bluetooth, cellular connections, or Zigbee protocols designed for low-powered processors [48]. IoT applications often connect to cloud servers for data storage, processing, and analysis [1]. Traditional security solutions need to be modified to address new challenges in the IoT paradigm.

Intrusion Detection Systems (IDS) have been used with centralized machine learning to address security concerns in conventional internet models. There is significant interest in developing new archetypes of these existing security systems to address the IoT challenge. One area of recent interest is Federated Learning (FL). FL is a new paradigm of machine learning that mitigates storage capacity limitations and data privacy concerns for the ubiquitous IoT sensors with heterogeneous resource limitations.

FL was introduced in 2016 by Google to address machine learning where data was unevenly distributed over a large set of nodes [21]. However, federated learning models present two significant challenges pertaining to IDS for IoT. One challenge is that most IoT networks create non-independent and identical datasets, Non-IID, due to their ubiquitous and heterogenous nature. Non-IID is a problem because it causes model accuracy to decrease by 55% [19].

The second challenge is that federated learning is vulnerable to adversarial attacks. One

attack that significantly affects federated models is the poisoning attack. A poisoning attack is when a malicious client entity can provide local training weights that corrupt the global model located on a centralized server. See Figure 1.1: Data Poisoning in Machine Learning below. On the left side of the figure the honest client data at the top of the image is filtered into the machine learning model. There the data is processed and trains the machine. The output is a model with a successful learning rate.

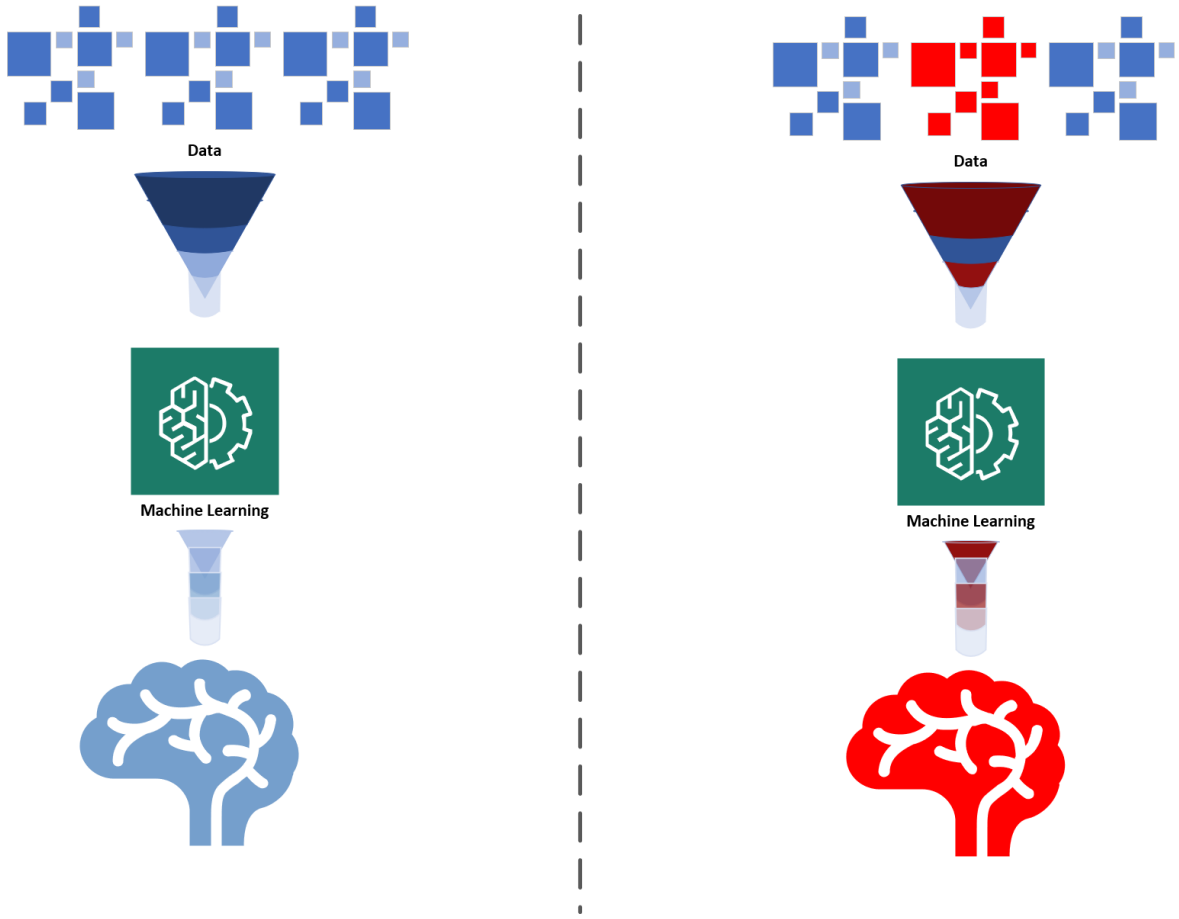

Figure 1.1: Data Poisoning in Machine Learning

On the right side of figure 1.1, poisoned data in red is utilized by a malicious client. The

data is accepted by the model for processing and training. The end result is a model that fails to learn or meet desired convergence metrics.

This has created an urgent need to develop poisoning defenses. There exists a broad gap in poisoning defenses in relation to intrusion detection systems and federated learning models. The few defenses from centralized models have framed poison identification as either a two or three dimensional classification problem to be solved with spatial metrics to contrast honest clients from poison attackers. These defenses have missed critical information that should be included into the problem framing for poison attacker identification. Poisoning attacks exist in random sequences of time. This creates the need for a four dimensional solution that uniquely solves individual classification prediction from a random sequence over a given time period.

This *ST-DSD* report contributes a four dimensional neural network that can accurately receive input from the time dimension as an additional feature when comparing local model updates through the lens of spatial distance similarity. This is the first report to utilize a Bi-Directional Long-Short Term Memory (Bi-LSTM) Recurrent Neural Network (RNN) with a deep stacked TimeDistributed Dense layer allowing more efficient random sequence predictions in that four dimensional manifold. Additionally, we contribute four attack methodologies for poisoning a federated intrusion detection system. Finally, we explore with the need for future work an anomaly based federated IDS for IoT using a similar deep neural network. To the best of our knowledge, this is the first report that examines poisoning attacks in a federated IDS for IoT that frames the problem through the lens of spacetime.

1.1 Background

1.1.1 Internet of Things

IoT architecture can be generalized in a 3-layer model: application layer, network layer, and perception layer [48], [1]. IoT devices vary from sensors, RFID tags, actuators, smart phones, and other devices. These devises often use one or more of the following technologies:

Ipv6, RPL, 6LoWPAN, UMTS, Wi-Fi, and Bluetooth. The result is that IoT networks are heterogenous themselves. Figure 1.2: 3-Layer IoT Architecture Model demonstrates this generalization.

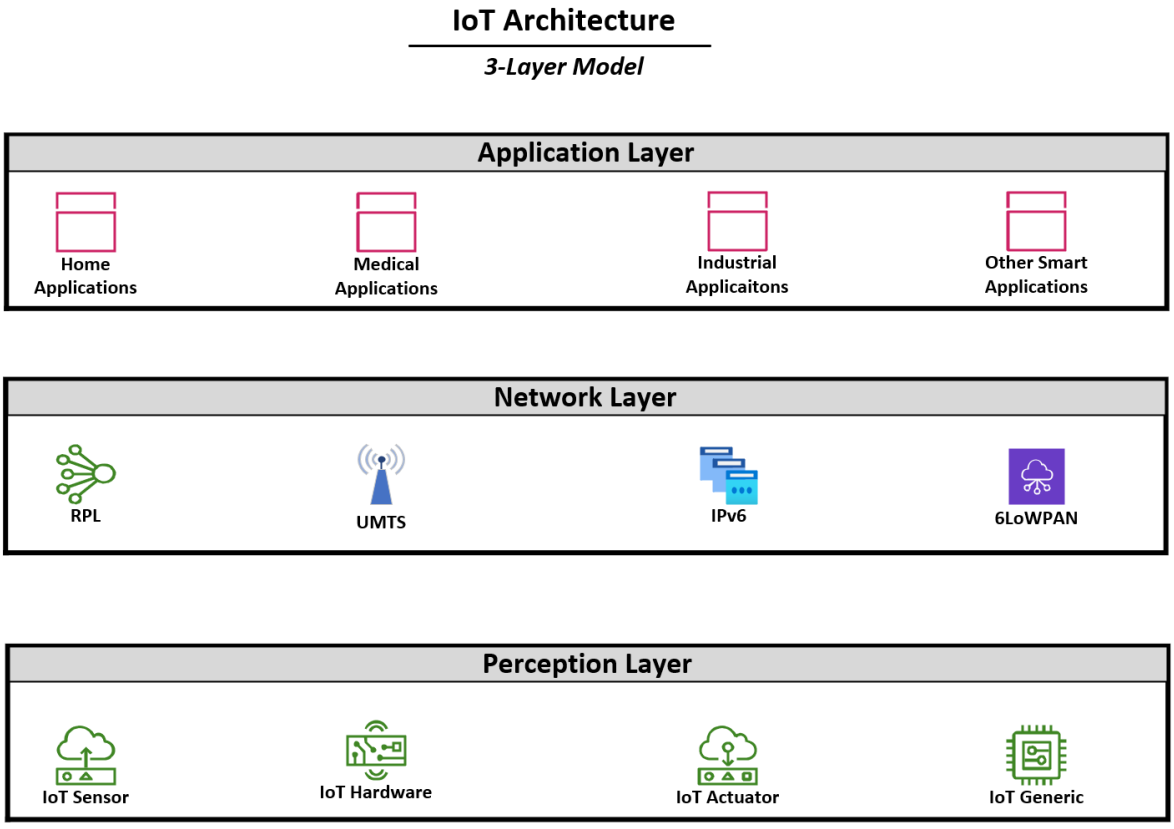

Figure 1.2: 3-Layer IoT Architecture Model

The top layer of the figure demonstrates the various applications that have driven IoT device saturation in our current ecosystem. One of the most important conlcusions from this figure comes from Network Layer and Perception layer. The network layer consists of many different protocols that allow for different communication architectures by the multifarious devices in the perception layer. The varying nature of IoT devices and their network deployments creates new challenges for the IoT paradigm. Privacy and security are two of those

challenges facing this architecture [31].

1.1.2 IoT Security

IoT Security (IoTSEC) presents a unique set of problems due to the growth of IoT popularity, and the diversity of deployment configurations in IoT networks. The Congressional Research Service (CRS), a nonpartisan shared staff, predicted the expansion of IoT sensors from 9.9 billion in 2019 to 21.5 billion in 2025 [13]. This doubling of IoT devices will provide ubiquitous targets in critical national infrastructure, industrial (IIoT) settings, medical (IoMT) settings, smart cities, and smart homes. IoT networks are complex often creating cross-device dependencies which expands the attack surface of the IoT network [50]. This increased attack surface aggravates the security concerns related to the pervasive presence of sensors. The result of IoT expansion and the rise of these security concerns is the need to examine mechanisms to secure this unique ecosystem.

Traditional security practices are insufficient for IoT devices. The modern security ecosystem consists of static perimeter network defenses (Firewalls), end-host defenses (Antivirus), and software patches [50] as seen in Figure 1.3: Traditional Network Security Model.

IoT devices disrupt the static perimeter defenses due to their omnipresence and the cross-device dependencies. Further, host-based solutions do not address the limited resources available to the IoT devices. Finally, IoT devices are routinely delinquent of routine patching protocols [31]. These challenges require adapting traditional security practices to address the disruption caused by IoT.

1.1.3 Intrusion Detection Systems

Intrusion detection systems (IDS) are one mechanism that has helped secure traditional networks by monitoring systems or networks for malicious activity or policy violations [3, 20, 34, 38, 40, 41, 43, 44]. In recent years, machine learning is increasingly used to build reliable and more effective intrusion detection systems. However, the IoT security challenges present unique hurdles for IDS using machine learning. One of the issues in IoT is that traffic and

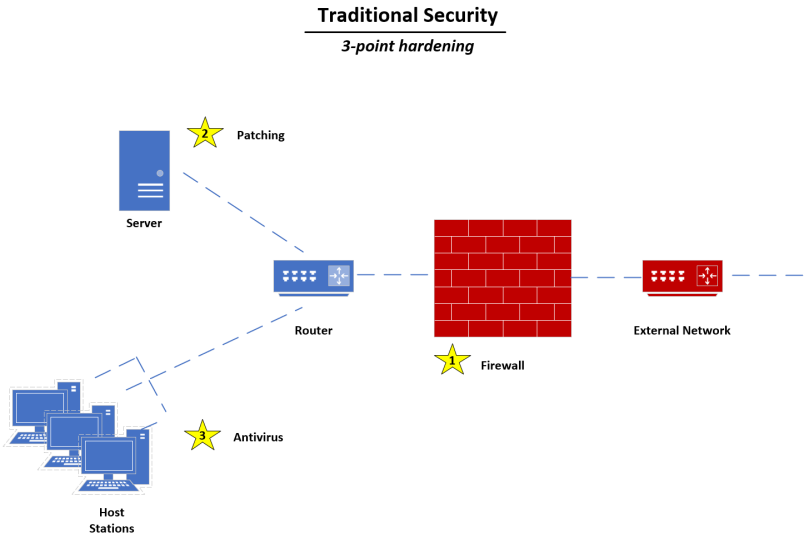

Figure 1.3: Traditional Network Security Model

data is created at the edge of the network. The networks are not centralized and there are no guarantees of data distribution or independence. Further, the storage capacity of low powered IoT devices limits the data that can be stored on the device for data processing. Since these devices are low powered, computationally intensive security measures such as cryptography cannot be directly applied to secure IoT environments. This is a challenge that requires the machine learning intrusion detection archetype to adapt for IoT.

1.1.4 Federated Learning

Federated learning is an adaptation that shows promise in overcoming the distribution and independence challenges created in IoT. FL creates a centralized model that prevents data leakage and improves security by keeping the training data private to the individual nodes. One area incorporating FL is intrusion detection [9, 23, 25, 29, 30, 32, 36, 37, 49]. This is demonstrated in Figure 1.4: Federated IDS with IoT Clients. A centralized global model accepts inputs from the pervasive IoT nodes, and then aggregates the individual weightings updating the distributable version.

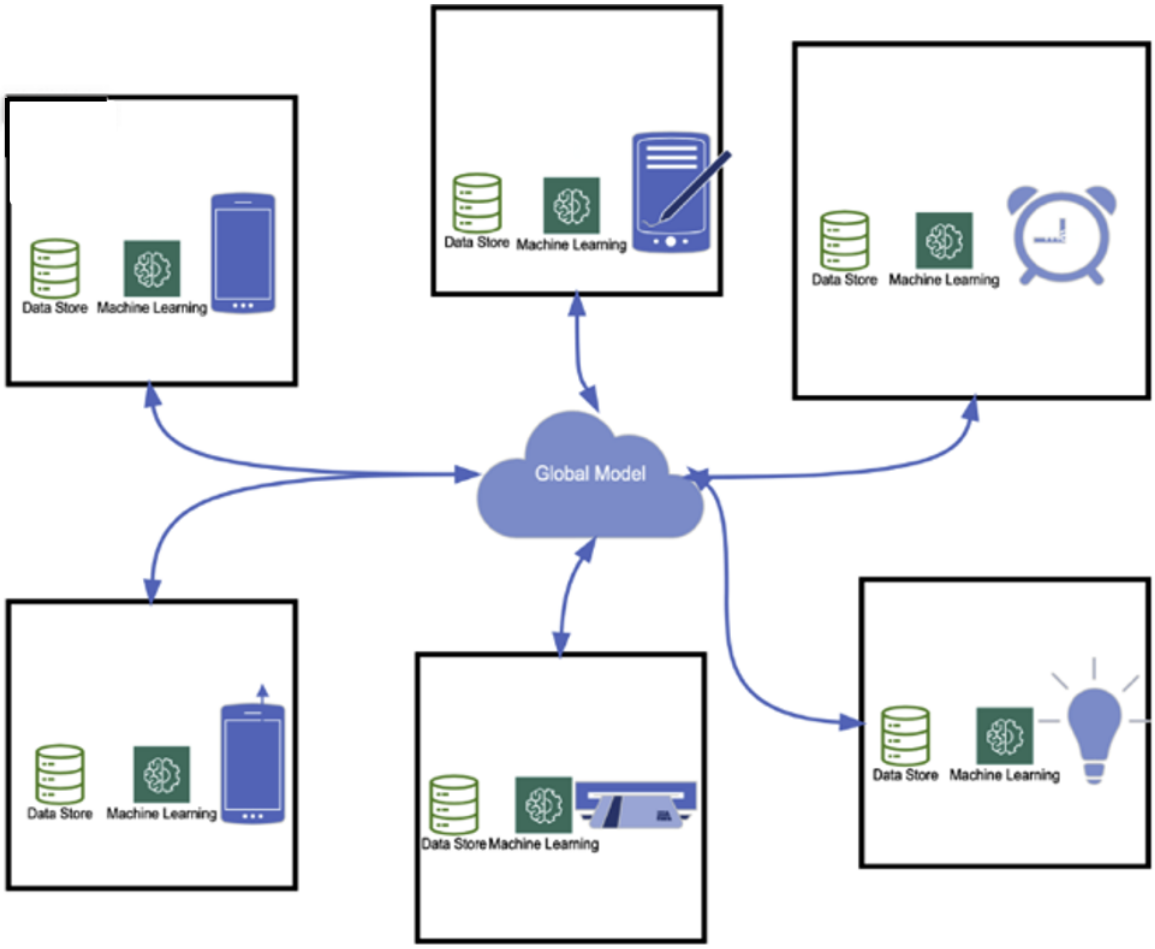

Figure 1.4: Federated Learning Model with Internet of Things Clients

One hurdle that must be overcome for FL IDS is the vulnerability to poisoning attacks. A federated model allows the local client to train the global model with local data and upload the updated weights to be aggregated. There exists an attack vector for malicious actors to upload corrupted models to negatively impact the convergence of the global system.

1.1.5 Poisoning Attacks

Byzantine and Sybil attacks are two types of poisoning attacks. Both attacks may use multiple attackers, however they differ in their end goal. Byzantine poisoning attacks cause convergence failure of the final model. Achieving optimum performance is defined as convergence. Convergence is most often measured by model loss. Model loss is a penalty for incorrect predictions. The penalty can be applied through many formulas such as mean squared error. When a model achieves convergence with 50% of the participants being malicious, it is considered Byzantine resistant [26]. Byzantine attacks are constructed by implementing workers who behave randomly or are untargeted attacks. In The random input of the Byzantines degrades the model. Sybil methodologies do not attempt to prevent convergence of the global model. Instead, Sybils focus on targeting the convergence output of certain classes of the model [14]. Therefore Sybil attacks are considered targeted attacks.

There exists limited research on addressing such adversarial attacks on the system [2, 10, 16, 22–24, 27, 39, 42, 45, 51, 52]. Poisoning defenses have primarily focused on imaging data sets in traditional ML models. This thesis researches the defense impacts against poison attacks on a novel anomaly-based intrusion detection for IoT using federated learning. First, we construct a federated anomaly-based IDS for IoT, and then we devise several poisoning attacks to affect the global weights for different gradients. We then implement a bidirectional LSTM Recurrent Neural Network to examine spatial relationships between the different local gradient updates. Our LSTM implements a many-to-one regression architecture that examines several distance based pairwise similarities between the vectorized gradients of each local model batch through a dimension of time.

Chapter 2

RELATED WORKS

The existing solutions for poisoning defenses generally utilize an individual similarity measurement. Each solution has demonstrated some strengths of that measurement with respect to specific attacks of that study. However, the existing solutions do not present a suitable defense against the modern evolution of data poisoning attacks.

2.1 Poisoning Attacks and Defense

Current published work for poisoning attacks is primarily limited to centralized learning paradigms [4, 23, 25, 32, 37, 39, 49, 51]. However, due to the promising results of early FL studies, there is a growing awareness of the vulnerability that FL has to adversary machines corrupting the global model with localized training weights [42]. A second study verified FL vulnerabilities in two poisoning attacks: label flipping and backdoor attacks [51].

In response, there is a small but growing base of poisoning defense research. [51] and [6] studied Euclidean Distance in Sybil an Byzantine attacks. Next, cosine similarity related to targeted training set attacks produced a state of the art defense in [14] and [16]. There have been several studies that utilize cryptography to secure FL models from poisoning. [23], [22], and [27] all explored defenses using Single key, Homomorphic Encryption, and Multi-party computation. Despite the growing interest in poisoning attacks, there exists very limited information on poisoning attacks in a federated learning environment for IoT intrusion detection.

2.2 Similarity Defenses

Federated poisoning defenses present many gaps and assumptions due to the infancy of the federated learning (FL) paradigm. Poisoning attacks create mathematical signatures due to their effect on the machine learning model. These measurements are depicted in Figure 2.1: Various Similarity Measures in Earlier Defenses. Specifically, the product of global aggregation from the local weights creates these mathematical similarities. Recent studies have examined the global model's measure of change through cosine similarity, Euclidian distances, triangle area similarity and sector area similarity(TS-SS), and logits to identify poisoning attacks [52], [16], [18]. Hammoudeh and Lowd assumed that by focusing on cosine similarity gradients, their attack agonistic approach would apply even to zero-day attacks [16]. However, the authors in [47] were capable of defeating the gradient-based poisoning defense. Additional studies [15] and [35] also defeated gradient-based defenses with a distributed backdoor poisoning attack.

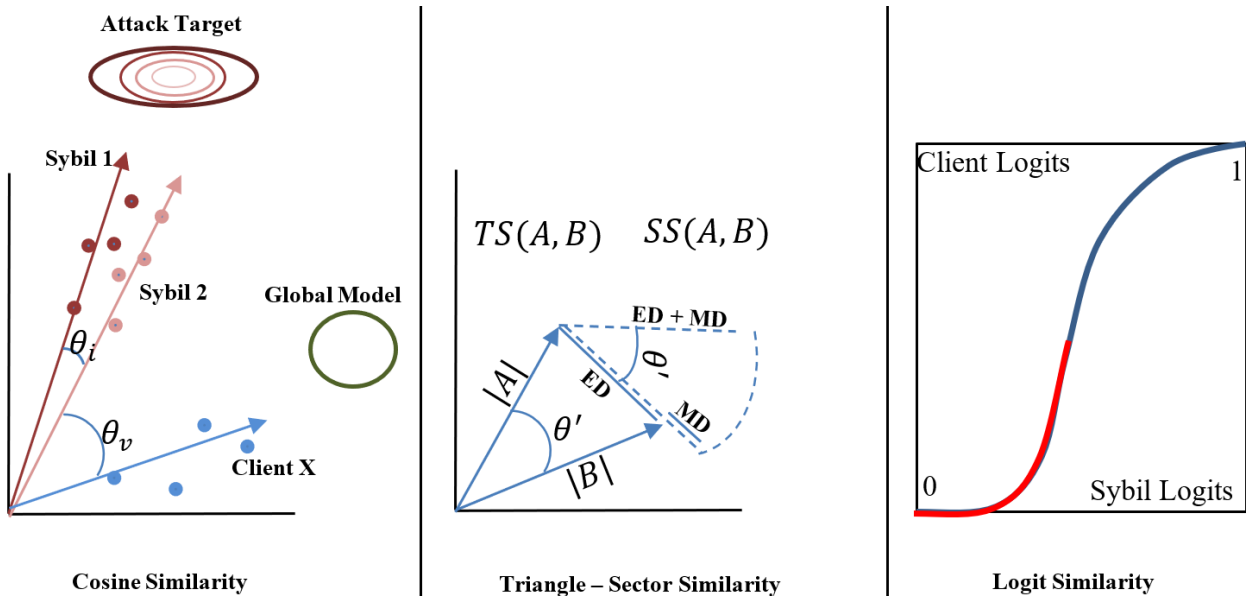

Figure 2.1: Various Similarity Measures in Earlier Defenses

Heidarian and Dinneen focused on Euclidian distance and magnitude differences to produce better purity in document clustering similarities with the TS-SS algorithm [18]. This work was then adapted for novel instrumentation by Birchman to provide resistance to distributed backdoor attacks in an algorithm called Area Similarity FoolsGold – ASF [5]. Birchman's work incorporated Heidarian's triangle area and sector area similarity work into the FoolsGold algorithm. ASF replaced the gradient defense with Euclidian distance and magnitude difference. The difference resulted in better functioning of the honest client pardoning. However, the implementation reduced the accuracy of identifying Sybils. Birchman still achieved a better convergence measured by loss when poisoned by DBAs, largely due to honest client pardoning.

Zhang et al. identified that the local models of attackers and benign participants had significant differences in the diversity of their output logits [52]. The FoolsGold algorithm also adopted logits; however, they were used to protect low-scoring honest clients with a gradient less than 0. When incorporated to identify poisoning attacks in all the studies, cosine similarity identifies Sybils most accurately. Birchman's ASF implementation of FoolsGold achieves higher model accuracy in DBAs. Finally, FoolsGold achieves the highest Sybil identification accuracy in label and backdoor attacks.

This research aims to achieve the backdoor Sybil identification strength of cosine similarity, the DBA convergence hardening of ASF, and the Byzantine prowess of multi-Krum in one state-of-the-art neural network. We assume that the non-IID nature of IoT networks includes spacetime relationships that require a deep learning archetype capable of utilizing human-like memory to identify spatial similarity patterns over time sequences provided by poisoning participants that can better predict these malicious attackers. That prediction can be increased if the model is able to propogate data from past to future, and from future to past. We present this model as *ST-DSD* and test it against various attacks in an IoT network.

Chapter 3

PROPOSED DESIGN

To study poisoning defenses we explored different poisoning attacks and implemented four novel attacks. Those attacks cover the historical evolution of poisoning attacks to the current model. Our thesis aims to provide a solution to these specific attacks. We propose *SpaceTime - Deep Similarity Defense*, *ST-DSD*, to secure federated learning models in an IoT environment from poisoning attacks.

3.1 Motivation

The primary purpose of this thesis is to explore mathematical means to harden federated IoT IDS models against poisoning attacks from malicious participants. The *ST-DSD* logic is based on three assumptions:

3.1.1 Poison Attacks shift towards multiple or distributed attackers

[14], [15], [5] all demonstrate a shift in focusing on the increased success rates created by distributed Sybil and Byzantine attacks as seen in [8] and also evidenced in the Multi-Krum defense [6]. This assumption encourages *ST-DSD* to examine spatial and time relationships between multiple attackers.

3.1.2 Multiple or distributed attackers create clustered signatures

[14], [51], [8], and [15] demonstrate clustering through different measures of spatial similarity. Sybil participants in Backdoor attacks exhibit significant gradient similarities. DBAs create a pronounced magnitude difference of individual gradients because they focus each attacker

on a single weight's convergence. *ST-DSD* includes gradient and distance-based similarity features to provide comprehensive protection to the various attacks.

3.1.3 Poisoning defenses for federated IoT IDS are a spacetime problem

The vectorized distances from the local batches can be described using Pythagorean theorem where $(\triangle D_{istance})^2 = (\triangle x)^2 + (\triangle y)^2 + (\triangle z)^2$. These are the first three dimensions in spacetime. Those spatial measures could be handled by most machine learning models. However, distributed attackers may be presented in different time continuum's. Let 'c' be the constant speed of light, 't' be time, and then the $\triangle ct = c\triangle t$ in ct-coordinate. This creates the three + one dimension of space time where $\triangle S_{pacetime}$ = spatial time. Then, spacetime is defined as $(\triangle S_{pacetime})^2 = (\triangle ct)^2 - (\triangle x)^2 - (\triangle y)^2 - (\triangle z)^2$. In the context of this research Figure 3.1: Participant Identification through spacetime Manifold graphically depicts this manifold.

The red clients are attackers and the blue clients are benign. The individual red and blue clients may or may not be, or have relations with, the same client in a different time frame. The cube represents time from the bottom row to the top. The number of clients is represented in the width. While graphically depicted as uniform, the number of clients could be jagged. The individual clients poses distance and time relationships. A Long-Short Term Memory Recurrent Neural Network can solve spacetime problems due to their architecture allowing the implementation of timesteps. The ability to include long-term memory may help a model contextualize points in the three + one dimension of space and time to identify attackers. This assumption drove this research to implement a novel bidirectional LSTM model to evaluate poisoning defenses in a federated environment through the problem description of spacetime.

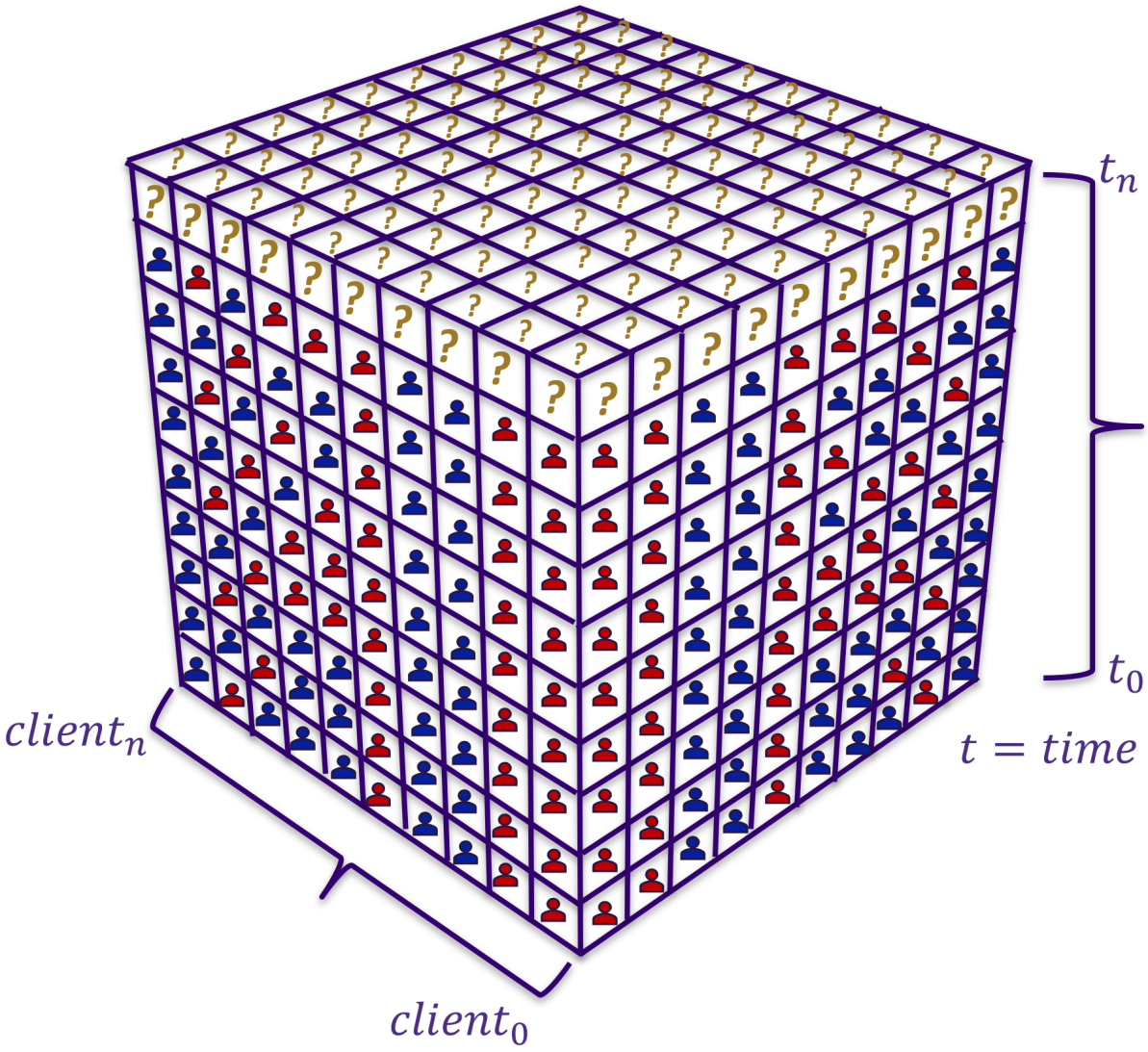

Figure 3.1: Participant Identification through Spacetime Manifold

3.2 Defense Model

3.2.1 Defense Architecture

This research builds an LSTM Recurrent Neural Network using the many-to-one regression architecture for poisoning detection based upon multiple mathematical similarities named

ST-DSD. *ST-DSD* integrates as a component to a federated IDS for IoT. Architecturally, *ST-DSD* is a micro-service that receives the local gradients prior to the global model. Figure 3.2: *ST-DSD* Cloud Deployed Architecture shows *ST-DSD* as the poison detection icon. After predicting the local weights, *ST-DSD* passes the honest client gradients to the global model for aggregation.

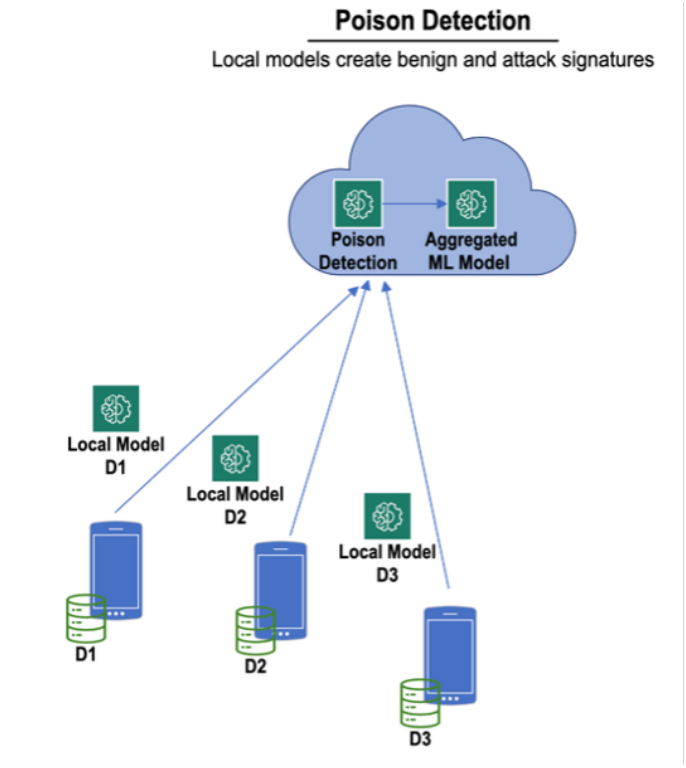

Figure 3.2: *ST-DSD* Cloud Deployed Architecture

ST-DSD computes five similarities, a density distribution, probability, and dispersion for each set of vectorized updated local gradients. The system extracts the product of triangle area similarity (TS) and sector area similarity (SS) referred to as Area Sector FoolsGold (ASF), cosine similarity (CS), Manhattan Similarity (MS), Euclidean Distance (ED), Jaccard Similarity (JS), normalized distribution, inverse logits, and standard deviation from each of the local models per update batch. *ST-DSD* then uses a stacked bidirectional LSTM model

with timesteps to create predictions. Next, *ST-DSD* passes that prediction to the aggregation algorithm of the IDS. The IDS will not aggregate the local models that *ST-DSD* predicts as poison attacks.

3.2.2 Defense Framework

The *ST-DSD* defense is based upon adapting theorems of similarity to identify poisoning attacks in federated learning IDS models. These are included as the features to our neural network. The distance between two points in a two-dimensional plane is known as Euclidean Distance. ED helps detect byzantine attacks. Let us consider the points $A(x_1, y_1)$ and $B(x_2, y_2)$ and let $ED = distance$.

Then ED is derived:

$$ED = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]} \tag{3.1}$$

Manhattan Distance (MD), is the distance between two vectors also known as city blocks. It is equal to the one-norm of the distance between the two vectors. MD is commonly used in regression analysis to find straight lines. MD can be derived through the sum of absolute differences. MD of $\vec{A}$ and $\vec{B}$ where $\vec{A}$ = [a,b,c] and $\vec{B}$ = [d,e,f]:

$$MD = |a - d| + |b - e| + |c - f| \tag{3.2}$$

The cosine similarity is the foundation of the defense as implemented in FoolsGold [15]. *Cosine similarity* is a metric that describes the cosine of the angle between two vectors projected in a multi-dimensional space. Cosine similarity can be used to describe likeness irrespective of magnitude. This is important in poisoning attack similarity as client-side hyper-parameters can modify magnitude.

When given two vectors, A and B, the cosine similarity, $\cos(\theta)$ can be represented as the dot product and magnitude where A_i and B_i are components of vector A and B:

$$\cos(\vec{A}, \vec{B}) = \frac{\sum_{i=1}^{n} A(i) \cdot B(i)}{|\vec{A} \cdot |\vec{B}|} \tag{3.3}$$

The resulting range is -1 to 1, whereas -1 means precisely the opposite, 0 indicates orthogonality, and values in between indicate similarity. Cosine similarity in poisoning attacks is graphically represented in Figure 3.3: Cosine Similarity of Sybils. The honest client vectors in blue push closer toward the global model and the actual objective, and they are dissimilar to the cos(θ) of the red Sybil gradients.

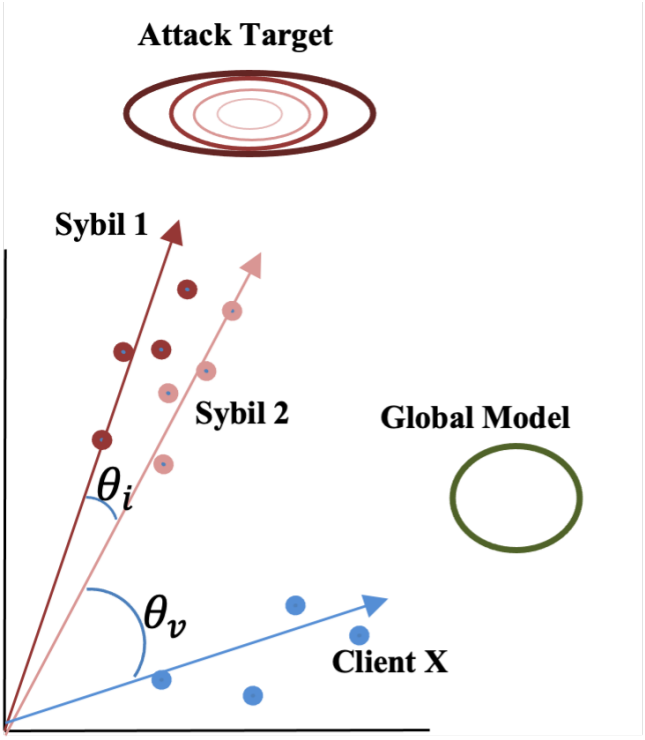

Figure 3.3: Cosine Similarity of Sybils

Triangle Area Similarity (TS) considers the angle, Euclidean distance, and the magnitude of a client's vectors. The authors of [18] argue that the resulting output is a more robust similarity metric. The TS logic is that two triangles are similar if they have the same shape, not necessarily the same size. This means their angles are equal, and their sides are proportional. If one of the two conditions, angle equality or proportionality, are true – the other condition is inevitably true.

Using the SAS (Side Angle Side) formula to calculate the area of a triangle includes the length of two sides and the Sin of the included angle. In this instance, A and B represent vector updates.

$$TS(\vec{A}, \vec{B}) = \frac{|\vec{A}| \cdot |\vec{B}| \cdot \sin(\theta')}{2} \tag{3.4}$$

The triangle area similarity equation decreases proportionally as the vectors approach each other, and the overlap of vectors will cause the formula to fail. Theta prime is determined by the arccosine of $\cos(\vec{A}, \vec{B}) + 10$ to prevent this edge case.

This minor adjustment prevents overlapping vectors from false readings where theta prime is determined:

$$\theta' = \cos^{-1}(\vec{V}) + 10 \tag{3.5}$$

However, TS accuracy can still fail when specific parameters for angle and Euclidean distance are met. Therefore, TS is not robust enough to produce accurate similarity due to some missing components.

Sector area similarity (SS) balances the triangle area similarity by considering magnitude differences in client vectors, and this allows for measuring an area between two vectors with a different perspective. The sector area similarity equation is the summation of magnitude difference and Euclidean difference squared with an angular rotation.

$$SS(A, B) = \pi \cdot (ED(A, B) + MD(A, B))^2 \cdot (\frac{\theta'}{360}) \tag{3.6}$$

Heidarian et al. proposed multiplying TS and SS in a TS-SS algorithm to provide clustering of documents based on similarity [18]. The TS-SS formula multiplies TS·SS and is presented where $TS - SS(A, B)$ is defined:

$$\frac{|\vec{A}| \cdot |\vec{B}| \cdot \sin(\theta') \cdot \theta' \cdot \pi \cdot (ED(A, B) + MD(A, B))'}{720} \tag{3.7}$$

TS-SS outputs a set from 0 to ∞ where 0 is only achieved when Euclidean distance is equal to Magnitude Difference (MD), where MD is:

$$MD(A, B) = |\sqrt{\sum_{n=1}^{k} A_n^2} - \sqrt{\sum_{n=1}^{k} B_n^2}| \tag{3.8}$$

The output strictly focuses on the relationship between Euclidian distance and magnitude difference. While the magnitude of a singular local weight may be a better defense against DBAs, as demonstrated by [5], label flipping and traditional backdoor attacks have shown affinity towards angle similarity value. During our defense methodology we refer to this formula as ASF due the implementation incorporating Birchman's adoption of the FoolsGold pardoning system.

Jaccard distance (JD) measures similarity of two sets based upon the number of observations in both sets. The range is from 0 to 1 with 1 indicating the same and 0 indicating no shared observances. The intersection is then divided by the union of observations in neither set.

Let A and B be vectors with local model gradients, then:

$$JD = \frac{|\vec{A} \cap \vec{B}|}{|\vec{A} \cup \vec{B}|} \tag{3.9}$$

The logit function is a quantile function in logistic distribution. Let p be the probability and expresses the logarithm of the odds. The product produces a probability expressed as elements of 0 and 1 mapped from $-\infty$ to ∞. The inverse logit is the transpose of this mapping. Let logit $(p) = \log(\frac{p}{1-p}) for p \epsilon (0,1)$. Then inverse logit (p) is defined:

$$(p) = \frac{exp(x)}{(1 + exp(x))} \tag{3.10}$$

Incorporating the logits for pardoning encourages a higher divergence for values at the two tails of similarity. This protects clients with low non-zero similarity values from Sybils by clipping the range of outputs as an element of (0,1). We implement this two tails effect with logits for each of the similarity measures. However, we also adapt inverse logits as a differentiating value between honest and attacker participants as evidenced by [52].

The normal distribution (ND) or bell curve is a continuous probability distribution. Understanding the gradient distribution of a vector may be of benefit in attacker identification and is given by:

$$ND(x) = \frac{1}{\sigma\sqrt{2 \cdot \pi}} \exp -\frac{1}{2}(\frac{x - \mu}{\sigma})^2 \tag{3.11}$$

Finally, the standard deviation (SD) measures the amount of variation or dispersion for the set of values:

$$SD = \sigma = \sqrt{\frac{\sum_{(x_i-\mu)^2}}{N}} \tag{3.12}$$

The dispersion of gradients could also provide our model with some clues to the origins of the participant. These 8 features were chosen for each participant to provide the Bidirectional LSTM with information about the three dimensions of space. Next we will discuss our methodology and implementation of the LSTM which allows for exploration of these dimensions through the perception of the dimension of time.

3.3 Defense Methodology

The *ST-DSD* defense begins by accepting the batch of local gradient updates prior to the FL IDS aggregation. ASF similarity is computed for each participant vector where participant vector X for participant $i = \vec{X}_i$. Let $\vec{X}_i \; \epsilon \; [\vec{X}_1, \vec{X}_2, \vec{X}_3, ...\vec{X}_i]$ where i represents the number of clients in the update batch.

Next, the pardoning algorithm from FoolsGold maps the ASF output to 0 and 1 for $\vec{X}_i$. This adoption is due to weak guarantees of similarities between attackers and honest clients. Clients are pardoned by re-weighing $\vec{X}_i$ and $\vec{X}_j$. The new client α_i is found by inverting the maximum similarity scores in the $\epsilon \; [0, 1]$. Finally; a logit function is incorporated to push Sybils and clients towards the two tails of divergence.

The final product is a weighted batch vector $B\vec{V}_v$, with ASF similarities of each client. Let this weighted Vector be $B\vec{V}_{asf}$. $B\vec{V}_{asf}$ should identify any vectors with significant magnitude difference. The assumption is that this should increase the DBA hardening of the system. We continue this same process for CS, MN, ED, JD, inverse logits, ND, and STD. However, JD, inverse logits, ND, and STD do not incorporate the FoolsGold pardoning mechanism as it would be counteractive. We describe each of those batched vectors as $B\vec{V}_{cs}$, $B\vec{V}_{mn}$, $B\vec{V}_{ed}$, $B\vec{V}_{jd}$, $B\vec{V}_{logits}$, $B\vec{V}_{nd}$, and $B\vec{V}_{std}$. The respective features helping to identify backdoor Sybils, byzantine attacks, or other anomalies.

Next, we implement a bidirectional LSTM model. LSTMs are a sequential network that handles the vanishing and exploding gradient problems in neural networks [46]. When working at scale, the exploding gradient problem exists where significant error gradients accumulate causing extensive updates in the model weights. This makes the model unstable and unable to learn. The vanishing gradient is the opposite problem. The gradients become so small that they do not update the model and can stop any further learning.

There exists many different architectures for a learning model that could be used to solve the spacetime problem. Recurrent Neural Networks (RNN) many-to-one architectures solve problems where there are many inputs and one output. In this case, we have many similarities: ASF, CS, ED, MN, JD, logits, ND and STD; but we only have one output for each set of similarities. That output is ϵ $[0, 1]$ for poison attacker or honest client. This architecture lends itself well to classification problems.

LSTMs allow information to persist like human memory due to their cell structure. To achieve human like memory, LSTMs utilize densely mapped neurons or cells. Each LSTM cell consists of a forget gate, input gate, and output gate [7]. The forget gate decides if the information is needed from a previous timestep or if it should be forgotten. The input gate weighs the importance of new information. The output gate is the prediction.

LSTM cell architecture is demonstrated in Figure 3.4: LSTM Cell with Input Gates. The black arrows represent a vector that is traveling through the neural network. The first sigmoid layer in the bottom left is the forget gate. The sigmoid outputs to a domain of 0 and 1. If the item is 0 all of the weights are discarded. Next, the input gate uses a tahn layer that outputs to the domain of -1 and 1. To update the cell state the tahn output conducts a pointwise operatation multiplying it by a sigmoid layer. Finally, a similar process is conducted to determine the output to the next cell where a pointwise tahn operation is multiplied by another sigmoid layer.

This allows us to understand the individual neurons that allow human like memory to persist in the network. Each vector weight has the ability to be deemed important and carried forward. Some items are deemed unimportant and are forgotten.

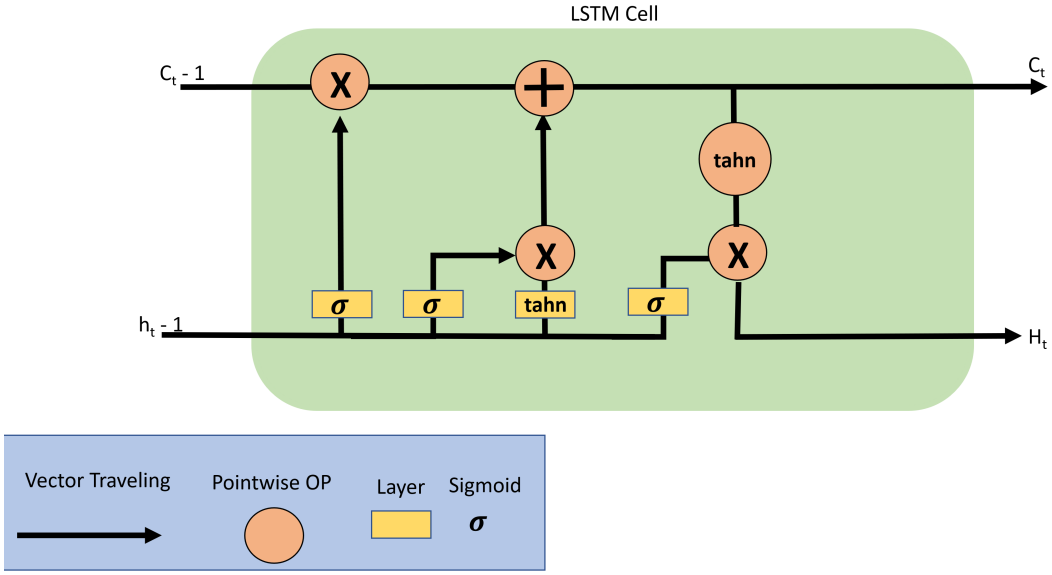

Figure 3.4: LSTM Cell with Input Gates

While LSTM cell structure provides human like memory, the Bidirectional LSTM provides non-human capabilities around the perception of time. We will use word prediction and a book metaphor to explain Bidirectional LSTMs. Memory allows us to read a book and make inferences in the current words we are reading due to the previous chapters as stored knowledge. A Bidirectional LSTM allows us to simultaneously read that same book from the end backwards to the word we are predicting. Thus, we are now able to use the persistent memory of words before and after the present word to make the prediction. Let us use the example:

[. . .In the evening the kids left the ?.]

We want to predict the question mark word. This is a low probability prediction without more context. Imagine now, in the Bidirectional LSTM we know the sentence after our

prediction is:

[They bought several clothes and other items.]

This additional context gives many clues to allow a higher probability prediction. Using a Bi-LSTM can increase the learning rate of the model at the extra computational costs of the additional layer. However, if that Bi-LSTM model is on a cloud-based server it should not impact the IoT network.

To put this back into the context of predicting poisoning participants, our *ST-DSD* model is reading a book from past to present and future to past at the same time. That book describes each client according to the similarities described in our framework. In this example, *ST-DSD* may have read the same JD score and dispersion for three of the last five participants. Additionally, as a Bidirectional LSTM it simultaneously reads that this JD score and dispersion is not in the next 12 participants. This additional context allows our model to make a prediction that the similar clients are attackers based upon these similarities and their relationship to the mapped probability of their vectors in the logit feature.

We can examine this in Figure 3.5: Bidirectional LSTM Forward and Backward Propagation. The bottom of the image contains the inputs for each timestep. Those inputs are used in both the forward and backward propagation layers. The input is acted upon by the LSTM cell and the output is carried forward to the next cell. The backward propagation outputs to the activation layer which results in the prediction at the top of the figure.

At any moment in time, or timestep, the model has persistent memory from past to present and future to present. To understand this concept better we must discuss the role of the timesteps. Timesteps allow each set of LSTM cells to utilize the information from the $N_{timesteps}$ previous (or future) LSTM cells. First let us assume training data and the value Y to be the output ϵ $[0, 1]$. The 0 marker corresponds to a poison client and 1 belongs to an honest client.

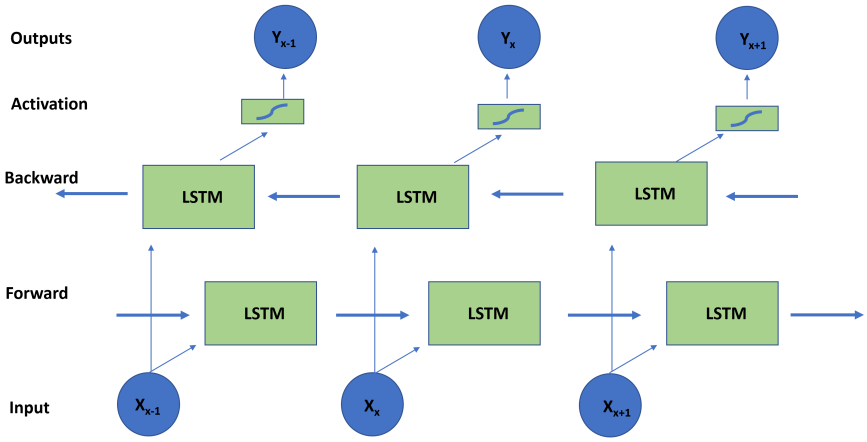

Figure 3.5: Bidirectional LSTM Forward and Backward Propagation

Let $\vec{V} = [B\vec{V}_{asf}, B\vec{V}_{cs}, ..., Y]$

Let $N_{timesteps} = 3$

Then:

$$Timestep\ 1 = [B\vec{V}_{asf}, B\vec{V}_{cs}, ..., 0]$$
$$Timestep\ 2 = [B\vec{V}_{asf}, B\vec{V}_{cs}, ..., 0]$$
$$Timestep\ 3 = [B\vec{V}_{asf}, B\vec{V}_{cs}, ..., Y]$$
$$Timestep\ 4 = [B\vec{V}_{asf}, B\vec{V}_{cs}, ..., 1]$$
$$Timestep\ 5 = [B\vec{V}_{asf}, B\vec{V}_{cs}, ..., 1]$$

This allows the model to examine all the information in Timesteps 1-3 with the assigned LSTM cells in the forward propogation layer to predict the output of Timestep 3. At the same time, the bidirectional component allows the LSTM model to read the information

from Timestep 5 to 3 in the backwards propogation layer to find any correlations to assist the prediction.

The *ST-DSD* Deep Neural Network Architecture in Figure 3.6 depicts the implentation for a backdoor attack with 20 Sybil participants. The Bi-LSTM model includes one neuron for each client in the time series at the top of the figure. This layer creates the forward and backward propagation sequence of clients. Next, the Bi-LSTM layer is mapped to 120 TimeDistributed Dense layers. TimeDistributed layers conduct convolutions with the returned outputs on each of the sequences. Finally, this layer condenses to a single TimeDistributed Dense layer to allow harvesting of the predicted output with a sigmoid activation mapping to the set $\epsilon\ (0, 1)$.

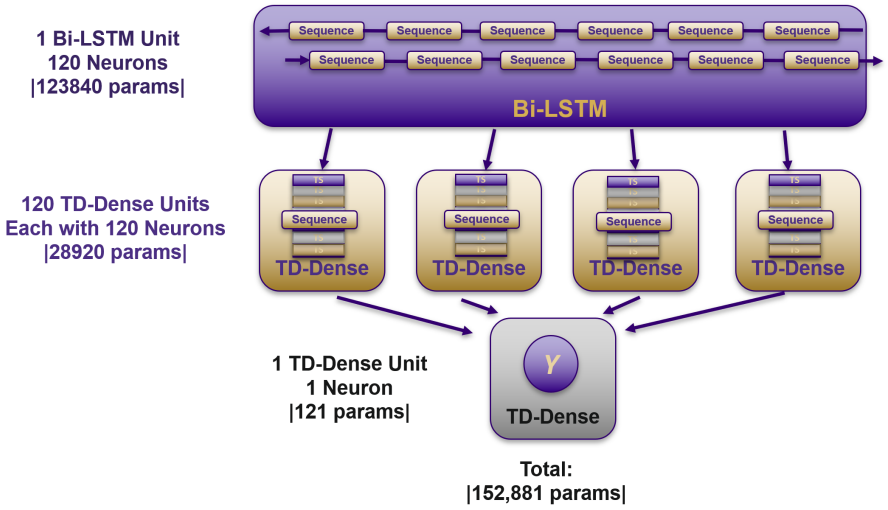

Figure 3.6: *ST-DSD* Deep Neural Network Architecture

The final output of the *ST-DSD* is a vector the length of of the number of clients in the update batch. Let that vector be $\vec{St}$ and let the prediction of each participant in the batch update be P_x. Then the output for $\vec{St} = [P_1, P_2, P_3, ..., P_x]$ for the length of the clients in the update batch. P_x is a value of 0 or 1 with 1 representing an honest participant and 0

being a poison attacker. When the IDS batch update aggregates the local model gradients, the algorithm is modified to omit nodes that map to the 0 prediction from *ST-DSD*. This methodology, the group of similarities, and the Bidirectional LSTM RNN is what separates the *ST-DSD* defense from [10], [16], [52], [18], [15], and [5].

3.4 Contributions

Our proposed design makes several contributions to the current work on poisoning defenses. To the best of our knowledge this is the first poisoning defense that explores the impacts of attacks on a federated anomaly-based IDS for IoT. We contribute a unique adversarial trained Bi-directional LSTM deep learning poisoning defense that utilizes spatial similarity and timesteps to identify poison attackers and pardon honest clients. We design and implement four poisoning attacks that utilize Byzantine and Sybil methodologies and are implemented with label flipping, backdoor, and distributed backdoor models.

Chapter 4

EXPERIMENT DESIGN

Our experiment set up a simulated federated IoT environment and implemented an anomaly-based IDS. We ran our tests on a host test machine built with a 32 core CPU and 4,325 cuda core GPU on a Windows 10 operating system using Visual Studio 2022 software. Each phase of the experiment simulated 20 clients training local models and sending them to the server for aggregation in 30 communication rounds. The data used simulated several attacks, honest clients did not manipulate the data. During iterations with attackers, the local data was manipulated to achieve the desired poisoning models objective. Data was collected during these attacks to provide an adversarial training set for our proposed defense. Our proposed design was trained on that adversarial training set and then incorporated into the experiment to evaluate its success.

4.1 Dataset

The dataset used for the IDSwas the UNSW Bot-IoT Dataset (Bot-IoT). Bot-IoT was created by designing a realistic network environment in a Cyber Range Lab at UNSW Canberra. The pcap files contain 72,000,000 records. The data selected for training originates from multiple CSV files where 535,051 abnormal or attack packets and 30,086 normal packets were curated. Attacks included DDoS, DoS, OS and Service Scan, Keylogging, and Data exfiltration attacks. Two separate datasets of 23,890 and 20,923 records was retained for testing.

From the 565,137 training packets, a second set was created to support the backdoor attacks. The intent was to create a backdoor gap for DoS attacks. This set was reduced to all packets that: 1) used TCP for the protocol, 2) sent to port 80, and contained 3) 1 packet

that was 4) less than or equal to 201 bytes. This set included 12,720 packets. This grouping then set the $TC_{'Attack'}$ for all packets that met these criteria to *Normal*. This poisoned data set was then used to create Sybils following the backdoor attack methodology. Four additional sets were created in a similar fashion for each single criterion 1-4 above. These data sets were used in constructing DBA attacks. These collective sets were used by the clients, backdoor Sybils, or DBA backdoor Sybils in the experimental methodology.

The dataset used to train the *ST-DSD* was generated by running the adverserial attacks against the IDS using its' training data and harvesting *ST-DSD* measurements. Over 120 convolutions produced 64,271 poison samples.

Table 4.1 is an example of the data contained in the training sets for the poisoning defense. It describes the similarity features from left to right. ASF stands for the Area Sector Fools Gold implementation of triangle area sector area similarity. FG represents FoolsGold's cosine similarity implementation. MN refers to Manhattan Distance. ED describes the Euclidean Distance. Inv Log is the inverse logit of the client vector. JD is the jaccardian distance. ND is the normalized distribution and STD represents the standard deviation of the vector. The far right column marked 'Y' is the label. Zero represents poison participants and 1 represents honest clients.

ST-DSD Data Set								
ASF	FG	MN	ED	Inv Log	JD	ND	STD	Y
0	0	1	1	0.1709	0	0	1	0
0	1	1	1	0.1709	0	0	0	0
1	0	0	0.4608	0.1714	0	0.7881	0	1

Table 4.1: *ST-DSD* Data Training Set Example

4.2 Intrusion Detection System Architecture

An anomaly-based IoT IDS using federated learning was constructed using a sequential LSTM model with 256 timesteps in 6 layers. The LSTM stack resulted in 3,290,113 parameters. Table 4.2 IDS Model Summary describes the layer type, shape, and parameters.

Anomaly-Based IoT IDS		
Layer Type	Output Shape	Number of Parameters
Dense	(None, 256, 256)	7680
LSTM	(None, 256, 256)	525,312
Dense	(None, 256, 512)	131,584
LSTM	(None, 512)	2,099,200
Dense	(None, 1024)	525,312
Dense	(None, 1)	1025
Total		3,290,113

Table 4.2: *IDS Model Summary*

Each local model was trained using one epoch on this IDS model to reduce the impact on IoT sensor resources. Finally, the model was fit with a batch size of 16 and a validation split of .2. This resulted in an average client training time of 7 seconds.

4.3 Threat Model

Poisoning attacks are adversarial attacks on ML models that allow the attacker to gain access to the learning of the system [12]. This access allows them to change classifications. By changing the classifications, the attacker misleads the training model to predict the target label 'TL' on any input data that the attacker has injected into the dataset [11]. These

attacks can be Byzantine random workers who disrupt the model or intentional Sybils who aim to avoid detection allowing the model to converge. Poisoning attacks are comprised of two main poisoning methods: label flipping and backdoors [51].

Label flipping attacks involve injecting crafted attack points into the training data by flipping the target class labels. Flipping the label will cause the trained model to adjust the original prediction boundaries. A Sybil label flip attack on an IoT IDS with two traffic classes ϵ ('Attack' , 'Normal') is accomplished by intentionally crafting attack labels with normal labels. To place Byzantine attacks into context, this same method could randomize all labels for both classes confusing the learning rate of the model.

Backdoor attacks differ from label flipping in the difference of their target. Backdoor attacks target specific attributes to make any class with that attribute mislabeled. The targeted data is referred to as the 'trigger' in a backdoor attack. Assume a DoS attack has specific destination port, protocol, and packet size information. Then, labeling all IoT traffic as normal that matches those criteria would be the trigger.

Label flipping attacks and backdoor attacks have one important difference. The significant distinction between the two attacks is that the backdoor poisoning attack fits the changed behaviors of the local model in a manner that the global model behaves normally on untampered data while achieving a high success rate with the injected samples [47].

The authors in [47] have already modified the backdoor poisoning attack to fit a distributed attack model. This modification is known as a distributed backdoor attack or DBA. DBAs have already become disruptive to early poisoning defenses. DBAs use additional Sybils to reduce the backdoor attack so that each Sybil targets a specific adjacent data point in the local gradients. In the DoS attack, one Sybil could poison HTTP protocol packets while other Sybils poison packets with less than 200 bytes. This granularity reduces the DBA Sybil's signature and similarity to other DBA Sybils. The final effect is that a DoS attack will converge to a 'Normal' label while evading current defenses.

4.3.1 Attack Methodology

Label flipping, backdoor, and DBA attacks were constructed with attacker participants ϵ $(1, 5, 10)$. For DBA attacks, each participant ϵ $(1, 5, 10)$ is a participant group of 4 attacker participants. The number of attackers were chosen to provide a complete range of attack rates across the four different attacks. Label flipping and backdoor attacks include attack rates of 5%, 25%, and 50%. The Distributed Backdoor Attacks were constructed with initial full data set attack rates of 20%, 40%, and 60%.

The following label flipping methodology was used to design both a Byzantine attack and a Sybil attack. In the Sybil label flipping methodology each Sybil set poisoned the target class label, TC_l, where $TC_l = 'Attack'$. Each TC_l can be described as $TC_l \cup local_{IoT_{packet_x}} \epsilon$ $('Attack')$. The Sybils then flipped the label to be $TC_l = ('Normal')$. Each Sybil generated a local model with their updated weights. The local weights were sent back to the global model during communication rounds. This is demonstrated in Figure 4.1: Sybil Label Flipping Methodology. The Sybil label flipping attack was designed to create a failure of the $('Attack')$ class in the model, while still achieving global convergence.

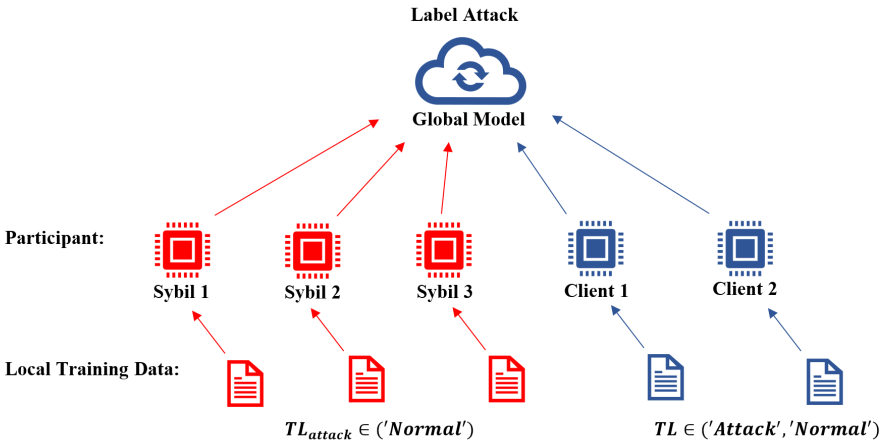

Figure 4.1: Sybil Label Flipping Methodology

The Byzantine attack was crafted in the same manner with exception to the differences described here. Each Byzantine set ϵ $(1, 5, 10)$ *randomized* each $local_{IoT_{packet_x}}$'s output label. Thus, each TC_l, where $TC_l = 'Attack'$ or $'Normal'$ was then generated through random permutation. The intent was to craft a vector with such large distance differences that when adopted by the model it would cause instability and significantly reduce global accuracy. The differences are highlighted in Figure 3.3: Byzantine Label Flipping Methodology.

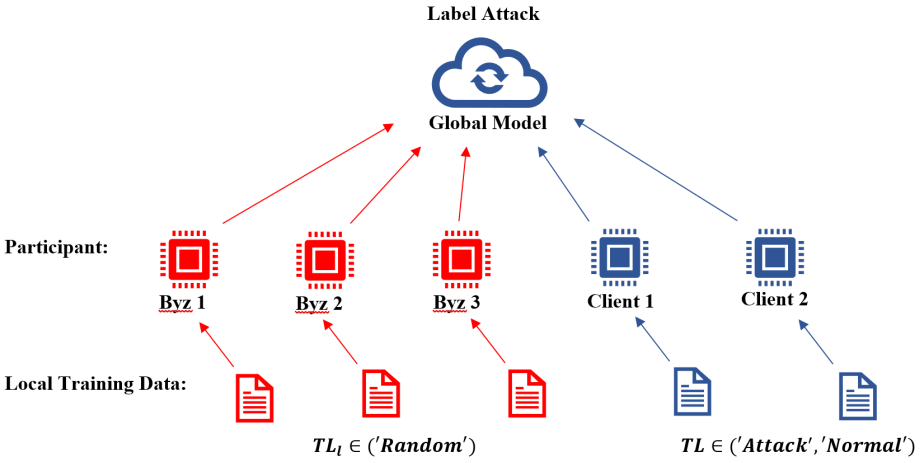

Figure 4.2: Byzantine Label Flipping Methodology

Comparing Figure 4.2 and 4.3, the global model at the top of both figures receives input from the participants. Participants contain honest clients and Sybil or Byzantine attackers depending on the implemented attack. All clients train on local data. However, the Sybil attackers corrupt their local data such that all IoT packets that contain the Target Label 'Attack' converge to 'Normal' where as the Byzantines in Figure 4.3 randomize the output of all the labels. This draws importance to the different objectives for the two separate attacks.

Next, the Backdoor methodology was constructed by creating Sybil participants ϵ $(1, 5, 10)$. Each Sybil set targeted the same set of features $\cup$ $local_{IoT_{packet_x}}$. That set of features were the trigger deemed to model a DoS attack. To construct the trigger let Local Data be described:

$$\text{LD} = \{local_{IoT_{packet_1}}, local_{IoT_{packet_2}}, \cdots, local_{IoT_{packet_i}}\}$$

Next, let $local_{IoT_{packet_i}} = |\vec{local_{IoT_{packet_i}}}|$; then:

$$|\vec{local_{IoT_{packet_i}}}| = \{feature_1, feature_2, \cdots, feature_x\}$$

The IoT IDS global model assumes that for each $feature_x$ of $local_{IoT_{packet_x}}$ ϵ $('Attack' \cup 'Normal')$. Let the attack trigger $= feature_{target}$ and be described as $feature_{target}$ ϵ $\{feature_1, feature_2, \cdots, feature_x\}$.

Thus, the goal of the backdoor Sybil is to change the convergence of the $feature_{target}$ of $local_{IoT_{packet_x}}$ ϵ $('Normal')$. This also implies the inverse that $feature_{target}$ of $local_{IoT_{packet_x}}$ $\notin$ $('Attack')$ for the DoS trigger. Figure 4.3: Sybil Backdoor Methodology removes the abstraction of this methodology and describes the selected features for the trigger.

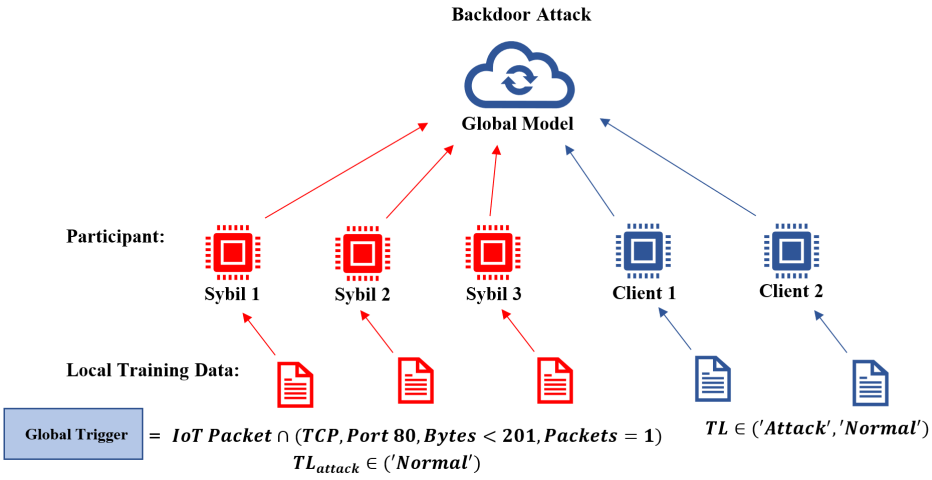

Figure 4.3: sybil Backdoor Methodology

Each Sybil in the backdoor attack contained a set of features from their packet data to

change the convergence of the global model to Normal. Those features were: 1) TCP protocol packets, 2) sent to port 80, 3) less than 201 bytes, and 4) consisting of 1 packet sent. This is considered the 'trigger' for the attackers. Each Sybil in the backdoor set changed the labels of all LD that met all of those four criteria. After training the local model with the corrupted data, the backdoor Sybil sent the updated weights to the global model.

The weakness of the above attack is that many Sybils are needed to affect the global weights when attacking a model at scale. That creates a very similar gradient for all backdoor Sybils and increases their ability to be identified.

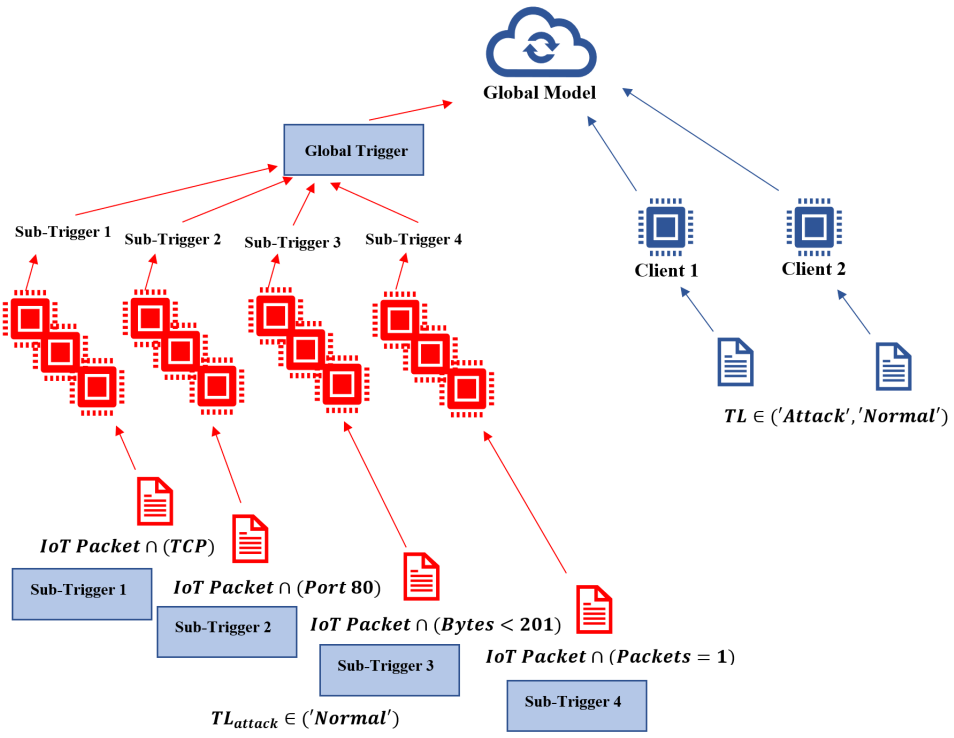

Figure 4.4: Sybil DBA Methodology

That weakness led to DBAs which avoid presenting a gradient signature. The DBA accomplishes this by separating each $feature_x \; \epsilon \; (feature_{target})$ to be attacked by each Sybil participant group $\epsilon \; (1, 5, 10)$. To again place this into simple terms, each of the features for

the backdoor attack –TCP, port 80, 201 bytes, and one packet; contained a set of Sybils targeting the convergence of just that feature or sub-trigger. The global trigger was still packets resembling DoS attacks demonstrated in Figure 4.4: Sybil DBA Methodology.

The DBA figure contains 4 separate sub-triggers each labeled as Trigger 1, 2, 3, and 4. Those sub-triggers correspond with one feature from the backdoor attack. In this case Trigger 1 are all TCP protocol packets. Trigger 2 is all packets sent to Port 80. Trigger 3 is all packets with a size less than 201 bytes. Finally, Trigger 4 are all packets with only one packet sent. Each Trigger contains it own set of Sybil attackers ϵ $(1, 5, 10)$ attempting to affect convergence of that data point. Each Sybil set attempts to change the TL for their trigger from 'Attack' to 'Normal'. Collectively the Sybil participant groups sum to ϵ $(16, 80, 160)$ and achieve convergence on the Global DoS trigger when all 4 sub-triggers converge.

This attack methodology assumes the attacker has white box knowledge of the IoT IDS data processing. This assumption aims to focus this work on IDS and poisoning hardening. Numerous works focus on implementation of poisoning attacks without knowledge [42], [51], [28], [33]. Future work to train the defense model should include those attacks.

4.4 Experiment Methodology

The experiment was conducted in a python environment utilizing cudaNN, TensorFlow-GPU, torch, and Keras packages. Four major steps were conducted: 1) IDS baseline; 2) Poison Data Generation; 3) Poison Training; 4) and *ST-DSD* Testing. Steps one, two, and four exercised the main code base with modifications to allow outputs that were desired for each step.

First, the IDS baseline was conducted in three iterations. The first test run demonstrated the IDS without any defense or attackers to produce baseline accuracy. The second experiment executed the four attack methodologies with each set of Byzantine or Sybil participants and no defense to demonstrate the attack impacts. During these attacks the code was modified to extract the similarity samples needed for adversarial training the *ST-DSD* model in

step 3. The third iteration tested the original FoolsGold implementation and the modified ASF defense to demonstrate results against each attack with all attacker permutations. The main program for all three experiments executed 2 *for loops* with an implicit 3rd *for loop*.

The first and *outer global loop* represented communication for each round to the global IDS model. This loop was conducted 30 times to simulate multiple federated updates from IoT participants. Prior to this outer global loop, all IDS data had been pre-processed, trained, and batched to the corresponding clients and attackers according to test parameters for that iteration.

Next, the second and *inner client IDS loop* created the local IDS models. It set the initial weights according to the global IDS model. Then this loop trained the local IDS models on each client's data individually and adjusted the local model weights.

The implicit 3rd *for loop* was the local epoch conducted creating the local weights. The new weights were scaled and then appended to the local weight list for each updated batch of clients. The program then moved back to the outer loop and summed the local weights from the list. During step one, iteration three, FoolsGold and ASF implementations were demonstrated prior to aggregation. In the fourth step, the *ST-DSD* component integrated prior to the aggregation of weights to make it's predictions and output it's mapped vector.

Step three was conducted separately, training the *ST-DSD* model on the generated attack data. Once the training was complete, the best epoch model was saved as a complete Tensorflow model that can be loaded into the IDS to complete step four. This allows *ST-DSD* to be a modular component added onto other federated learning modules for testing.

Chapter 5

EVALUATION

5.1 Evaluation Metrics

To understand our defense in context of current studies we compare our results to FoolsGold and ASF. First, we describe Byzantine and Sybil resilience through model convergence. Second, we analyze the precision of each model. Then, we discuss the poison rate and it's effect on a model's accuracy. Finally, we are testing models in the scope of intrusion detection and thus we will explore accuracy metrics in relation to these defense descriptions.

Convergence

Model error loss per epoch, or training round, determines the convergence of a system with different attack rates. Over time the loss of a model demonstrates a tendency of behavior. That tendency demonstrates the model has achieved it's capacity to learn. Loss can be used to explain overfitting and underfitting the model when comparing training loss and validation loss. However, this metric can be important in regards to attacks. Convergence describes a model's resilience during attack. We define poison resilience by achieving model convergence with 50% attackers.

Precision

Precision for this study consisted of two classes. Precision was measured for poison attackers and honest clients. Client precision is described as the percentage of clients correctly identified. Attacker precision is the percentage of poisoners identified as attackers. This metric is important because removal of honest clients, or allowing attackers to participate with their weights, has a negative impact on a federated learning model. Any defense needs to ensure

that both components of precision are as close to 1 as possible. This metric increases our depth of understanding why a particular defense struggled in a certain setting.

Poison Rate

The poison rate is the percent of model weights that were created by poison attackers. Poison rate should not be confused with attack rate. The attack rate is the total number of attackers attempting to poison a model. If a poison defense prevents an attacker from poisoning the model, that attackers data is not summed into the poison weight. Thus, in a model with all attackers and no clients, but a defense that identified all the attackers the attack rate would be 100% and the poison rate would be 0%. The poison rate shows the impact of an attack on the model accuracy.

5.2 Results

IDS Base Line

The IDS baseline conducted packet detection with no attackers and no defense to determine attack and normal packet traffic. There were 23,890 IoT packets examined between 20 clients. 9490 packets were benign traffic and 14400 were attacks. 7 different protocols were included: arp(467), icmp(12), igmp(2), ipv6-icmp(88), rarp(1), tcp(8447), and udp(14,873). The baseline was conducted in 33 communication rounds to provide additional information at the tail of the results. This led to 660 aggregated weights that updated the global model on the 33 iterations. One problem that was encountered later in the experiment was the negative impact of removing attackers on the neural network architecture for the IDS. The impact of removing clients demonstrated the fragility of neural networks in a small experiment and will be discussed under challenges.

In figure 5.1: IDS Baseline Accuracy with no Attacks we illustrate accuracy on the Y-axis through the lens of communication rounds on the X-axis. The graph begins the Y-axis at 0.55 to exclude data points not included in the results. The X-axis begins at 0 and ends at

32 representing the 33 communication cycles. The initial round demonstrates an accuracy of 59% and quickly climbs to 94.9% and beyond in the next rounds.

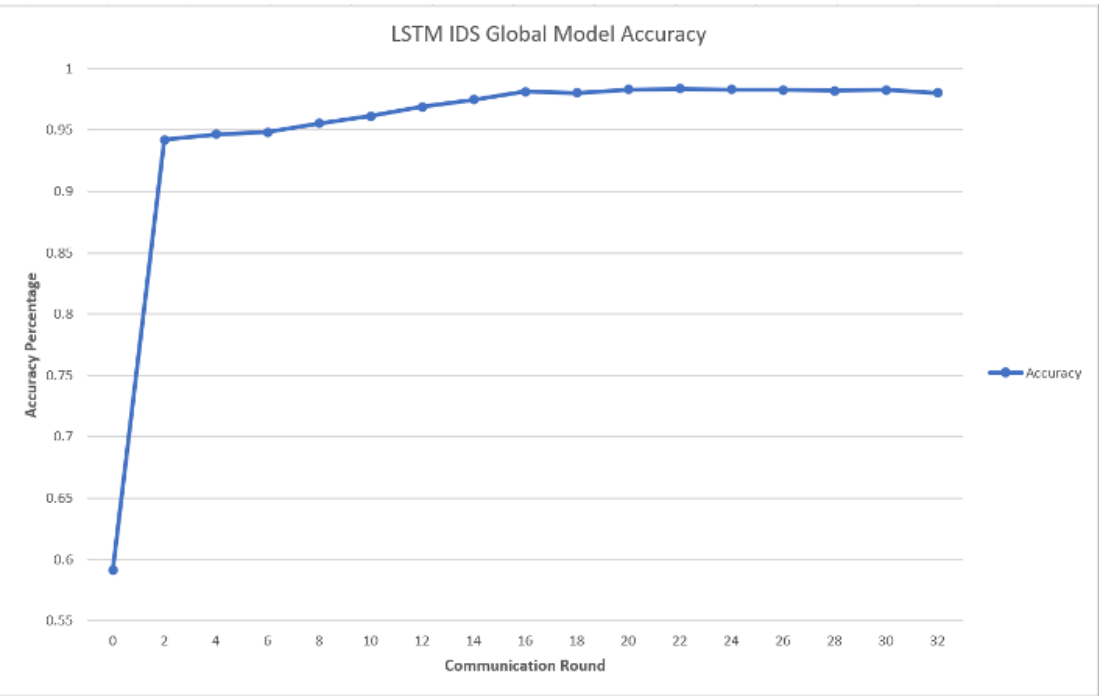

Figure 5.1: IDS Baseline Accuracy with no Attacks

In the 23rd round our anomaly-based IDS using FL for IoT demonstrates a maximum accuracy of 98.4%. The average accuracy over 33 rounds is 94.8%. The median accuracy of all the scores is 98.0%. We define IDS accuracy convergence as the median of last 15 communication rounds to allow an acceptable ramp up time for the federated model. The accuracy convergence is visually located at the point of inflection depicted in the curve at communication round 18. Using the last 15 rounds, our baseline accuracy convergence is 98.2%.

5.2.1 ST-DSD *Model Convergence*

Figure 5.2: *ST-DSD* Model Accuracy and Loss depicts the model loss on the left and accuracy on the right. The orange lines represent validation metrics and blue line represent training metrics. The Y-axis is accuracy or loss depending on the left side or right side of the figure. The X-axis lays out our training rounds. Our model has a loss convergence of 0.1 and an accuracy convergence of 0.99. We allow for 10,000 epochs to train the model. However, to prevent over fitting we save the best epoch model, and use a patience of 1,000 epochs. Once 1,000 epochs has been reached without improving loss, the model is done training and it loads the best epoch model. By using TimeDistributed ragged sequencing layers we reduced our convergence training from 15,000 epochs to 350.

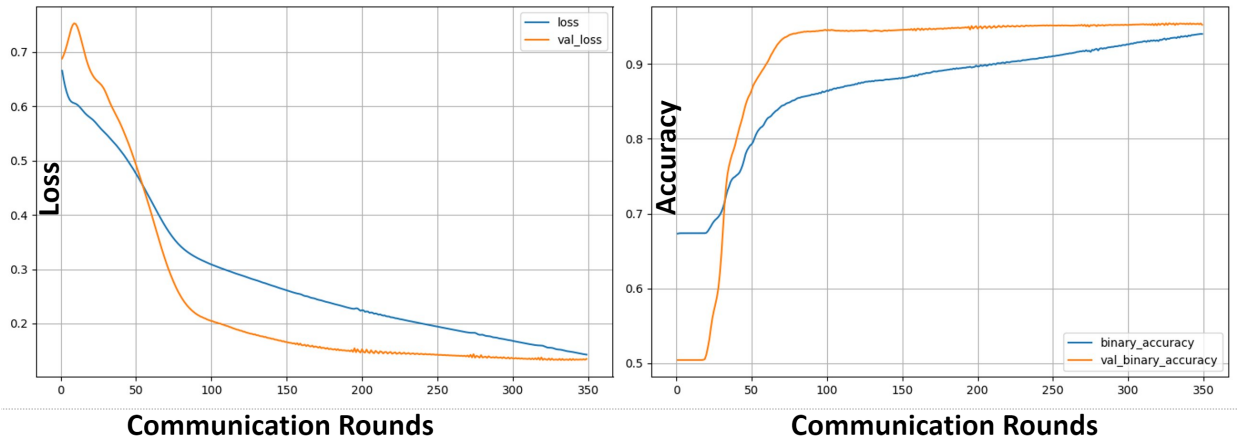

Figure 5.2: *ST-DSD* Model Accuracy and Loss

Poisoning Attacks

We examine the impact on the IDS baseline accuracy of 98.2% to measure the effectiveness of each attack methodology across different attack rates of Byzantine and Sybils ϵ $(1, 5, 10)$. Note that for the structure of DBA attacks, the participation is ϵ $(4, 8, 12)$. The number of honest clients was scaled to keep the batches to 20 clients. Thus, clients were ϵ $(19, 15, 10)$

with exclusion for DBA at ϵ $(16, 12, 8)$. We discuss the results in Figure 5.3 IDS Accuracy Impacts from Various Attacks in the section below. The Y-axis displays the accuracy from 50% to 100% and the X-axis shows our different attacks. For each attack we show the impact with 1, 5, or 10 attackers.

Our Byzantine attack was designed to cause model convergence failure. These attacks reduced the baseline accuracy to 59.1%. The 39.0% reduction in accuracy is considered a strong success. The attack was capable of affecting the global model convergence, which is a Byzantine end state. The backdoor Sybil attacks allowed the global model to achieve

IDS Accuracy Impacts from Various Attacks				
Attack Rate	Byzantine	Label Flip	Backdoor	DBA
5%	74.5%	83.3%	98.3%	99%
10%	64.4%	78.4%	98.2%	93.2%
50%	59.1%	64.1%	97.1%	89.1%

Table 5.1: *IDS Accuracy Impacts from Various Attacks*

convergence with both 1 and 5 Sybil attackers. The IDS baseline accuracy for all three attack rates was 98.3%, 98.2%, and 97.1%. The implications of these results is that at a 50% participant rate the attack has a 1.1% influence on the overall accuracy of the model. As an attacker, this methodology needs to be examined in greater detail to avoid detection.

The DBA attacks demonstrated similar impacts on the global model. At a 4 attackers we see the global model accuracy is 99.0%. However, the 8 and 12 attacker sets have significant impacts at 9.2% and 89.1% resulting in a 5% and 9% impact on global accuracy. These results could be caused due to the low scale of participants. The small size reduces the data ingestion of our neural network and adds a fragility to the model even with client update scaling.

From Figure 5.3 we can make the following assertions. First, the Byzantine attack graph-

ically demonstrates the model failure. Second, the label flip attack has a greater impact on the global model than we desired. A Sybil poisoning attack attempts to avoid detection by not impacting the global convergence. Third, the backdoor attack demonstrates a near perfect Sybil attack by not reducing the IDS accuracy. Finally, the DBA begins to show more impact on the global model at higher attacker rates.

5.2.2 Model Convergence

We can discuss convergence through the lens of both loss and accuracy. The individual clients demonstrated the greatest depiction of convergence through comparison. In figure 5.3: Model Convergence Demonstrated with both Loss and Accuracy we examine the IDS model convergence for an honest client over 8 epochs. The left figure draws the loss measurement over training cycles and the right graphic represents accuracy over the same periods. The orange lines represent validation data and the blue lines represent testing data. The left side, or Y-axis is marked in tenths representing either loss or accuracy. The X-axis represents 8 epoch training cycles. The loss lines are smooth and move from the top left to the bottom

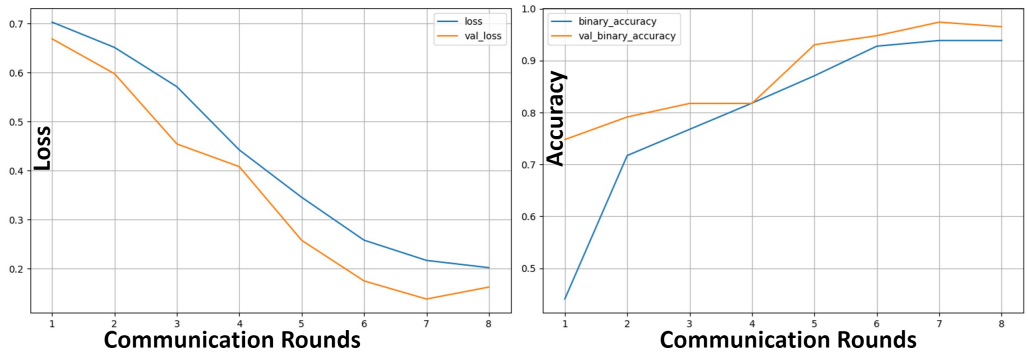

Figure 5.3: Model Convergence Demonstrated with both Loss and Accuracy

right, inverse of the accuracy chart. Notice that around epoch 7 in both figures there is an inflection point. This point could be considered convergence if the model stays near this level. The early callback function in Keras removes the additional data points to the right

of those lines that confirms these data levels stabilizing. The convergence for model loss is approximately 0.1 and accuracy is 0.98. Next, we examine a Byzantine poisoned model failing to meet convergence in Figure 5.4.

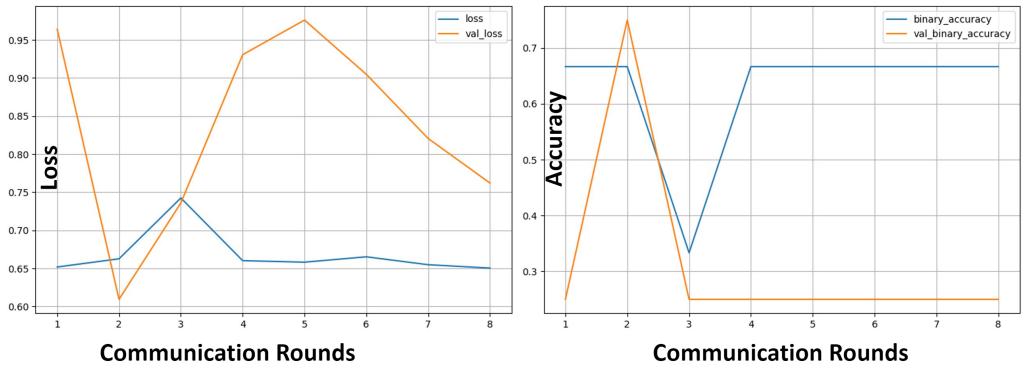

Figure 5.4: Byzantine Model Convergence Failure

The validation loss starts at 0.96 and varies greatly over the course of the epochs with a low reading of 0.62. Training loss has less variance but at 0.65 fails to meet the 0.1 convergence of the honest client in Figure 5.3. Accuracy shows a similar scenario. Validation accuracy is extremely low, and training accuracy at 0.69 fails to come within terms of the 0.98 convergence.

These convergence depictions are important because the failure identifies the success of the attack. Based on Figure 5.4 and 5.5 we can state that our implemented Byzantine attack causes a failure of the validation loss convergence by 51%. Additionally, we state that the accuracy convergence of the model is negatively impacted by 25%. This will allow us to determine the amount of impact our defense has in hardening the system against Byzantine attacks.

Poison Rate

In Table 5.2: Poison Rate Effects on Accuracy we highlight the reduction in a models accuracy as a correlation with the increase in the poison rate. Both attacks were taken

with a 50% attack rate. The table reflects the data from the 4Th communication round. This allows us to compare the FoolsGold (FG) defense against *ST-DSD*. The FG Attacker

Poison Rate Effects on Accuracy			
Defense	Attack Rate	Poison Rate	IDS Accuracy
FoolsGold	50	33	87
ST-DSD	50	0	96
Difference	0	33	9

Table 5.2: *Poison rate Effects on Accuracy*

precision was .5. That meant that 33% of the model weights were poisoned with DBA data. The effect was a reduction of 23% accuracy depicted by the blue IDS pie circling to 87% accuracy. In contrast,*ST-DSD* demonstrated a complete hardening at epoch 4. This achieved a 0% poison rate with a Sybil Attacker precision of 1 that corresponded with an IDS 96% accuracy for the round.

Poisoning Defense

We highlight our poisoning defense results by discussing the attack rate >= 50%. We chose this to demonstrate the resilience of the defenses. We include the precision of attackers and clients across the different attacks and defenses to provide comparison of results. We begin our discussion with Byzantine attacks and end with DBA.

In Figure 5.5: Byzantine precision Comparison at 50% >= Attack Rate. The Y-axis represents the total percentage of participants identified with 1 being the greatest. Across the X-axis each defense: ASF, FG, and *ST-DSD* demonstrate bar graphs of Attacker precision and Client precision. Holistically, the chart depicts a scarcity on the left side with fuller bars on the right.

During the Byzantine attack the ASF methodology had the poorest attacker identification

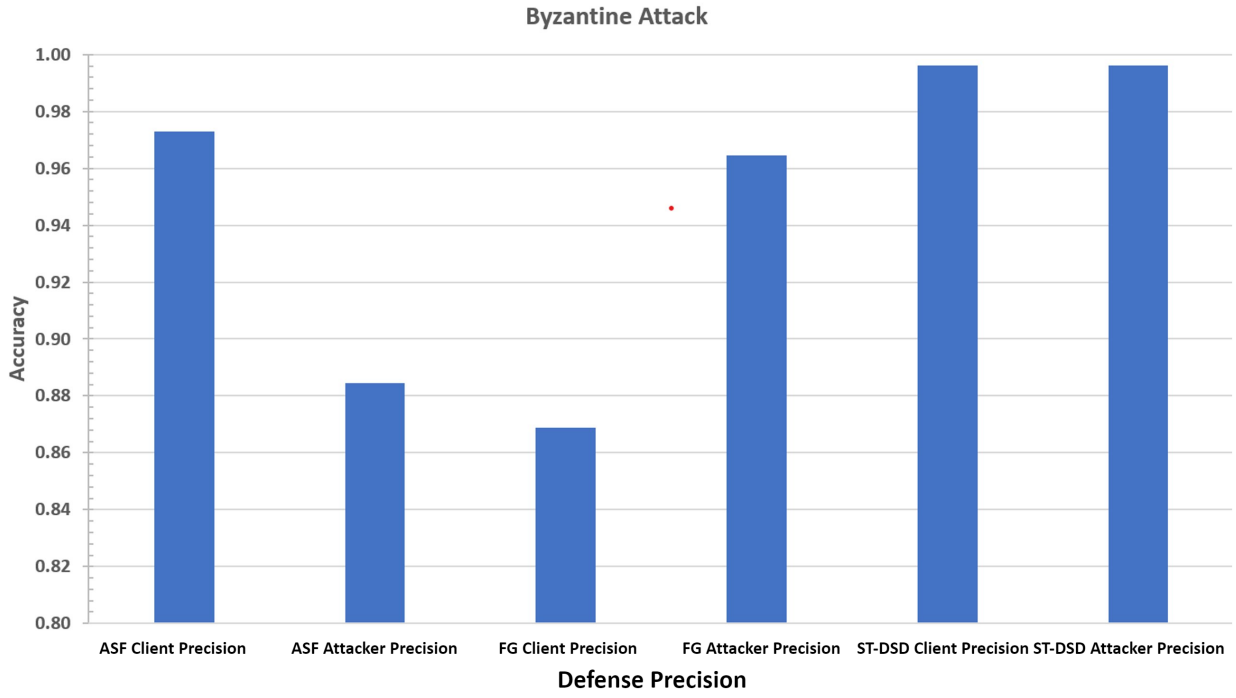

Figure 5.5: Byzantine Defense precision Comparison at 50% >= Attack Rate

with an attacker precision of 88.4% as depicted in Figure 5.5: Byzantine Defense Precision Comparison at 50% >= attack rate. FG had a decidedly better attacker precision at 96.4%. However, *ST-DSD* achieved the highest precision at 99.6%. The reason we examine precision is that inclusion of all honest clients is important for the IDS accuracy. We demonstrate the Byzantine client precision by defense in order ASF, FG, and ST: to be 97.3%, 86.8%, and 99%.

Next, the label flip attack in Figure 5.6 demonstrates comparable results from all defenses. The most worthwhile comparison was that ASF had a marked improvement on client precision in the label flip attack compared to the Byzantine attack. This indicates that ASF will handle Byzantine attacks poorly and is better designed to defend against Sybil attacks. FG client precision increased marginally, but was counteracted by a slight decrease in attacker precision. *ST-DSD* demonstrated both precision at 99%.

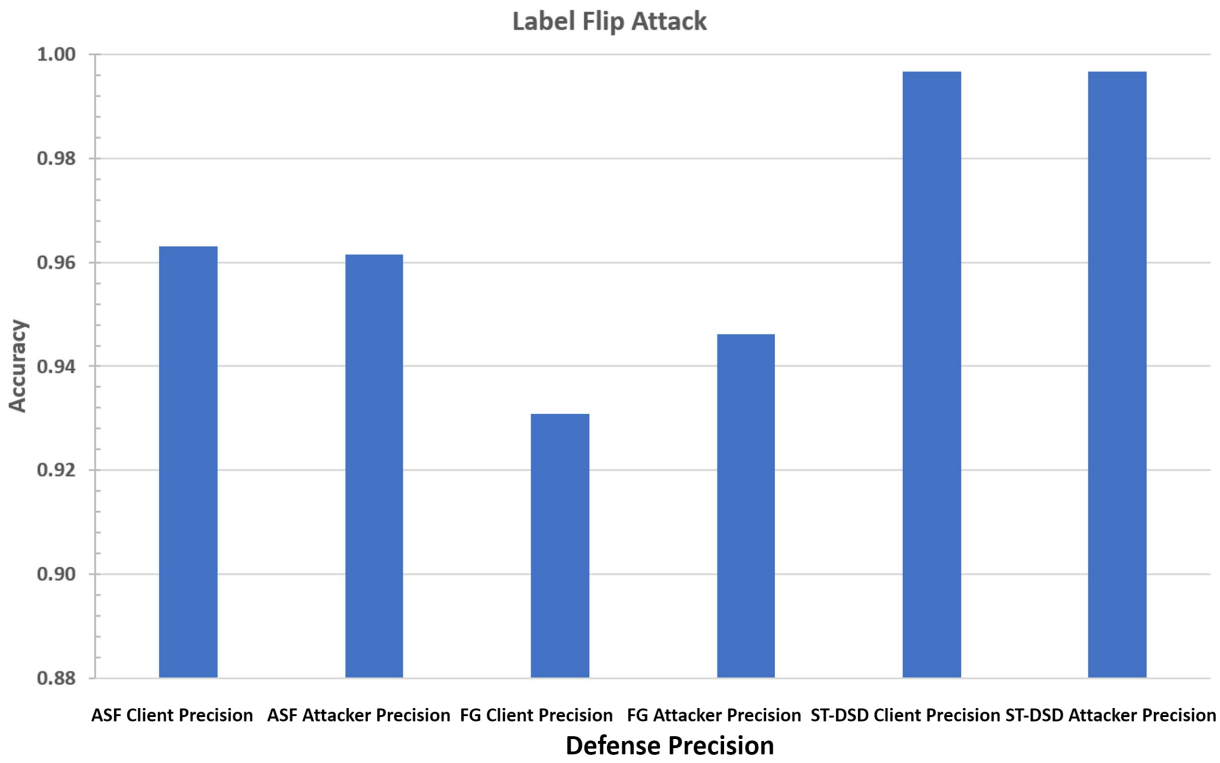

Figure 5.6: Label FLip Defense Precision Comparison at 50% >= Attack Rate

In Figure 5.7 the backdoor attack proved to more challenging for both ASF and FG. The backdoor attack rendered ASF statistically ineffective with a client precision at 74% and an attacker precision at 69%. FG demonstrated both spectrum's with an attacker precision of 100% and a client precision of 17.7%. FG was implemented to achieve attacker identification in backdoor attacks. These strong results reflect what is expected. What was not expected was the failure of the pardoning algorithm to prevent clients from being mislabeled as attackers. *ST-DSD* remained consistent with a client precision of 99.3% and 100% attacker precision.

The most evolved attack, the DBA proved to be the most challenging for all three defenses. Due to that challenge, we depict a collective precision of all three defenses against each of the attacks and invert the bar graph in Figure 5.8. This allows us to demonstrate how the

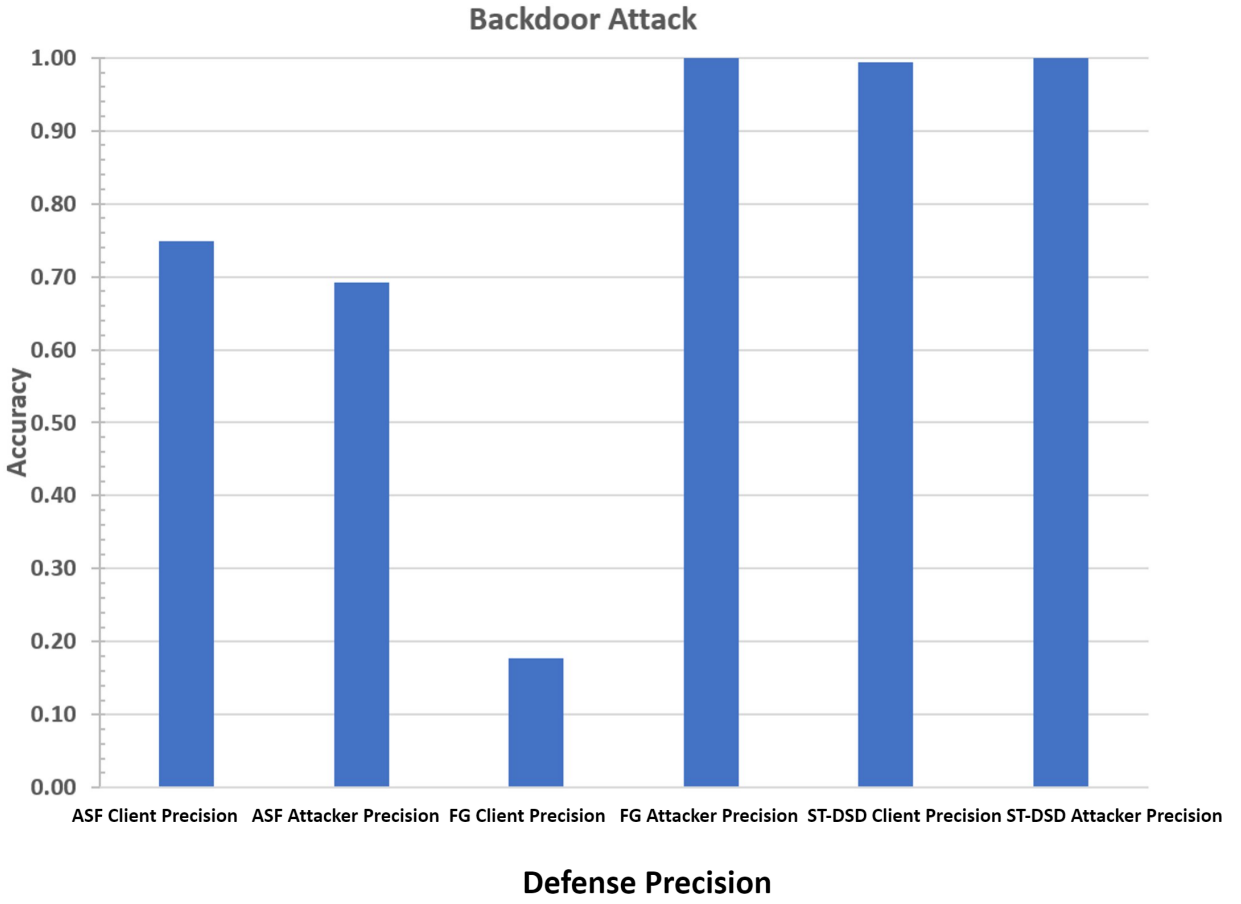

Figure 5.7: Backdoor Defense precision Comparison at 50% >= Attack Rate

DBA erodes precision compared to the other attacks.

First, the Y-axis represent the percentage out the 6 precision for the 3 defenses. A score of 6 is then mapped to 100% identification. A score of 0% maps to 0% and is not desired. Across the X-axis we follow the combined defenses successful identification rate across the Byzantine, label flip, backdoor, and then DBA attack. The figure depicts that Byzantine and label flip attacks are collectively identified with great success near 6. However, we notice that the percentage of precision for all three defenses slopes as we get to the backdoor attack at 4.5. Finally, the DBA presents the worst score 3.8%. One graphically important conclusion

is that the top two *ST-DSD* bars remain consistent in their width across all the attacks. This will be discussed next as we compare individual defenses.

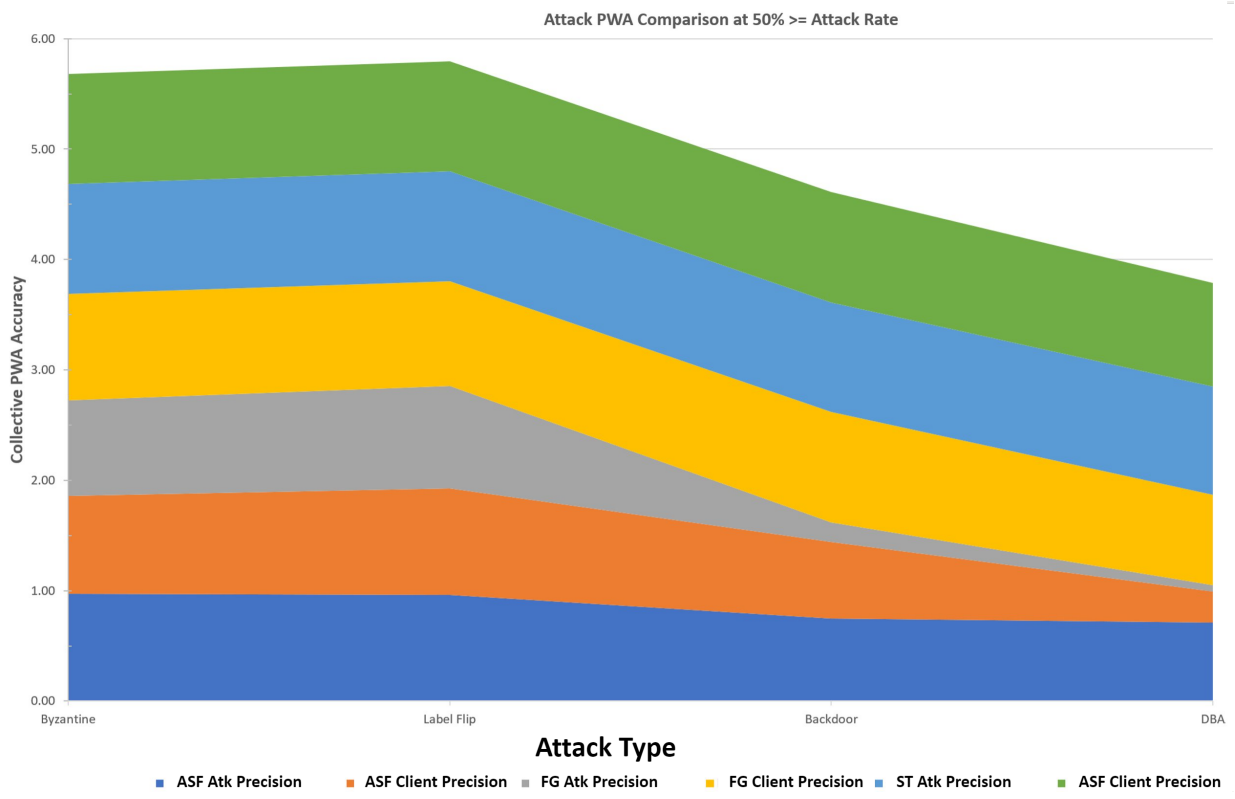

Figure 5.8: Collective Precision at 50% >= Attack Rate

We compare the individual defenses in Figure 5.9. The DBA attack reduced ASF and FG precision for clients and attackers to 71% and 28% for ASF, and 5% and 81% for FG. *ST-DSD* maintained a statistical significance with a client precision of 97% and an attacker precision of 94%.

We notice that ASF and FG demonstrate inverse relationships with client and attacker precision in the more challenging attacks. This may explain why ASF demonstrated higher IDS accuracy scores, as it included more honest clients. *ST-DSD* surpassed both FG and ASF in Byzantine label flip, Sybil label flip, Sybil Backdoor, and Sybil DBA attacks across all spectrum of attack rates.

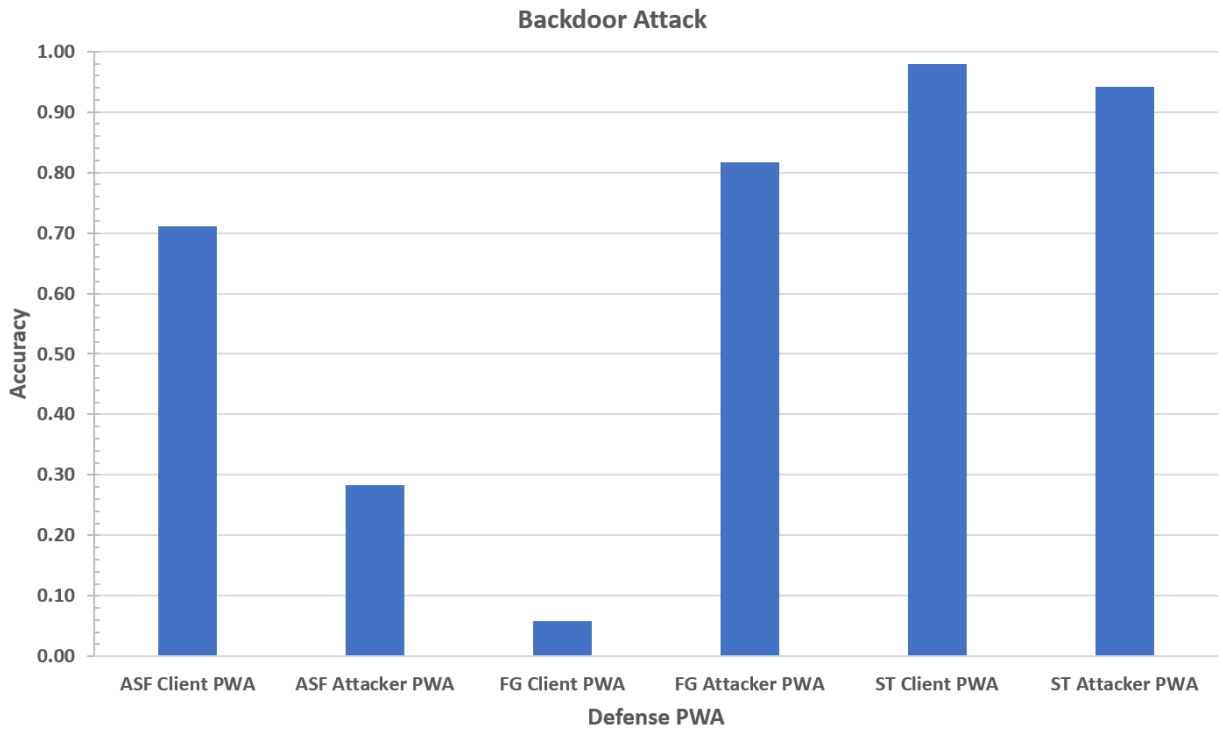

Figure 5.9: Individual Defense Precision against DBA at 50% >= Attack Rate

5.2.3 Problems with Results

Class distribution caused challenges for the IDS. IDS data contained 5% normal packets. To overcome these challenges class weights were used to increase the value of the under-represented samples. A second challenge was presented with architectural limitations of TimeDistributed dense layers. A TimeDistributed layer in this architecture is created to be a many-to-one RNN. Each client may send an unknown amount of packets. This is an issue for constructing a model with TimeDistributed layers. Implementing a batching protocol to standardize packet set size from clients would allow TimeDistributed classification.

The most significant challenge to the IDS was the fragility of the neural networks to removing data in the evaluations. Despite the near perfect removal of attackers, the IDS continually demonstrated lower accuracy as the sample set moved from 20 participants down

to 8. See Figure 5.11 IDS Diminished Accuracy which depicts the accuracy scores from the *ST-DSD* evaluations. Accuracy of the IDS is depicted on the Y-axis begining at 80%. The number of clients is depicted on the X-axis. What is demonstrated is that all attacks, despite near perfect attacker removal slow the learning rate of the model as there are less participants. A larger sample set should be evaluated to see if the impacts can be reduced.

Figure 5.10: IDS Diminished Accuracy

The Bot-Iot data set demonstrates the classic attack phases. The sequence of packets begins with reconnaissance and moves to other attacks. However, the data lacks non-attack packets. To create a manageable data set with a limited sample of normal attack packets that sequencing was disrupted. Harvesting the data without disrupting the sequence and demonstrating normal packet distributions would be beneficial for training the model.

The *ST-DSD* requires significant data. Data from various permutations of attacks needs to be generated to increase the robustness of the model. Further, high convolution rates and GPU usage over time caused the adversarial trained data at times to run into integer overflows and NaNs (Not a Number). These issues disrupted the training cycles or had a byzantine effect by adding noise to the data. Manual data cleaning and inline functions

were used to reduce the noise of data being displaced to the mean. However, this reduced the amount of data used for the experiment and challenged the architectural design of the poisoning model. To over come this, fixed participant sets of 20 were used to ensure the greatest amount of data for most experiments. Future work should generate enough data samples that a singular model can be robust enough to handle different participant lengths.

5.2.4 Implications

ST-DSD demonstrated state-of-the-art results. There is high evidence and confidence that poison participation can be identified using the spacetime manifold. *ST-DSD* demonstrated that earlier defenses did not frame the problem correctly and therefore missed out on an entire dimension of identification. However, those early defenses provided the features needed to ensure that *ST-DSD* incorporated the Byzantine resilience of Krum, Sybil identification of Fung's FoolsGold, and the DBA hardening through better precision by Birchman's ASF. This deep learning RNN managed to extract the strengths of the earlier systems into one.

ST-DSD has implications beyond the IDS. The modular architecture of *ST-DSD* can be incorporated into any IDS for IoT in a federated environment. As a singular component, most systems can be easily modified for adaptation. Restructuring the aggregation algorithm to only accept clients that are mapped to the *ST-DSD* output is all that is needed.

The four poisoning attacks demonstrated a successful implementation of poisoning attacks against a federated IDS. Crafting attacks to allow backdoors for more conventional attacks such as DoS provides a realistic example for how poisoning can be used as a tool to achieve greater means for an attacker. However, these attacks should be reconstructed to be black box attacks without knowledge of the model. Then, these attacks should be ran to generate adversarial trained data for the *ST-DSD*.

The slight degradation in the DBA 12 Sybil attack may be resolved by the inclusion of a new feature. To detect the magnitude difference of a single steps weight may be the key to identifying the missing attackers. We propose measuring the greatest magnitude difference between a clients steps, and then created another vector of the greatest magnitude difference

for each client of an update.

Finally, Additional work should be done to hash the identity of clients and incorporate a client history into the *ST-DSD* equation. This may help to increase a fairness of the federate model as clients may move in and out of a network. This would allow a greater extraction of information from the time dimension. Further comparison work can be done to examine the benefit of storing multiple modules that have been trained specifically for various participant lengths versus creating a master module for all lengths.

Chapter 6

CONCLUSION

We presented a unique similarity poisoning defense model that used triangle similarity, sector similarity, cosine similarity, Euclidean Distance, Manhattan Distance, Jaccard Similarity, inverse logits, normal distribution, and standard deviation to provide resistance to poisoning attacks. We modeled a SOHO IoT network and created Byzantine and Sybil IoT label-flip, backdoor and distributed backdoor poisoning attacks. We evaluated a baseline resistance by measuring global model convergence to a median of the last 15 communication rounds. We tested standard defenses adapted to an IDS for IoT. FoolsGold and ASF defense models were then used to make comparison to ST-DSD. Our defense was constructed using an RNN many-to-one architecture with bidirectional Long-Short Term Memory and TimeDistributed layers to allow a ragged sequence classification using the spactime manifold. We demonstrated that our state-of-the-art poisoning defense could achieve a statistical significance hardening against poisoning attacks by increasing IDS model accuracy, loss convergence, and precision across the spectrum of attacks and attackers with non-IID. Our Similarity defense achieved 99.9% attacker and honest client identification and laid the ground work for a new line of poisoning defense.

BIBLIOGRAPHY

[1] H. Alqarni, W. Alnahari, and M. T. Quasim. Internet of things (iot) security requirements: Issues related to sensors. *2021 National Computing Colleges Conference (NCCC)*, pages 1–6, Mar. 2021.

[2] L. Andrade-Arenas and J. A. Ramos-Romero. Analysis and prevention of iot vulnerabilities by implementing a lightweight ad-iot intrusion detection system model. *in 2020 IEEE Congreso Bienal de Argentina (ARGENCON)*, pages 1–4, Dec. 2020.

[3] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap. A supervised intrusion detection system for smart home iot devices. *IEEE Internet Things J.*, 6(5):9042–9053, Oct. 2019.

[4] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi. Mitigating poisoning attacks on machine learning models: A data provenance based approach. *in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 103–110, Nov. 2017.

[5] B. Birchman. Mitigating poisoning attacks against federated learning defense algorithms.

[6] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. page 11.

[7] H. V. K. S. Buddana, S. S. Kaushik, Pvs. Manogna, and S. K. P.S. Word level lstm and recurrent neural network for automatic text generation. *in 2021 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–4, Jan. 2021.

[8] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun. Understanding distributed poisoning attack in federated learning. *in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 233–239, Dec. 2019.

[9] G. W. Cassales, H. Senger, E. R. de Faria, and A. Bifet. Idsa-iot: An intrusion detection system architecture for iot networks. *in 2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, Jun. 2019.

[10] L.-Y Chen, T.-C. Chiu, A.-C. Pang, and L.-C. Cheng. Fedequal: Defending model poisoning attacks in heterogeneous federated learning. *in 2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec. 2021.

[11] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. http://arxiv.org/abs/1712.05526, Dec. 2017. Accessed: Mar. 15, 2022.

[12] J. Clements and Y. Lao. Backdoor attacks on neural network operations. *in 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov. 2018.

[13] P. M. Figliola. The internet of things (iot): An overview. *Internet Things*, page 3.

[14] C. Fung, C. J. M. Yoon, and I. Beschastnikh. The limitations of federated learning in sybil settings. page 16.

[15] C. Fung, C. J. M. Yoon, and I. Beschastnikh. Mitigating sybils in federated learning poisoning. http://arxiv.org/abs/1808, Jul. 2020. Accessed: Mar. 17, 2022.

[16] Z. Hammoudeh and D. Lowd. Simple, attack-agnostic defense against targeted training set attacks using cosine similarity. page 15.

[17] A. Harit, A. Ezzati, and R. Elharti. Internet of things security: challenges and perspectives. *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, pages 1–8, Mar. 2017.

[18] A. Heidarian and M. J. Dinneen. A hybrid geometric approach for measuring similarity level among documents and document clustering. *in 2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 142–151, Mar. 2016.

[19] H. Hellstrom, V. Fodor, and C. Fischione. Wireless for machine learning. page 21.

[20] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2:20, Dec. 2019.

[21] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. http://arxiv.org/abs/1610.02527, Oct 2016. Accessed: Mar. 13, 2022.

[22] Y. Li, G. Hu, X. Liu, and Z. Ying. Cross the chasm: Scalable privacy-preserving federated learning against poisoning attack. *in 2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–5, Dec. 2021.

[23] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Trans. Inf. Forensics Secur.*, 16:4574–4588, 2021.

[24] Z. Liu, N. Thapa, A. Shaver, K. Roy, X. Yuan, and S. Khorsandroo. Anomaly detection on iot network intrusion using machine learning. *in 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pages 1–5, Aug. 2020.

[25] S. K. Lo, Q. Lu, C. Wang, H.-Y. Paik, and L. Zhu. A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Comput. Surv.*, 54(5):1–39, Jun. 2021.

[26] L. Lyu. Privacy and robustness in federated learning: Attacks and defenses. http://arxiv.org/abs/2012.06337, Jan. 2022. Accessed: May. 04, 2022.

[27] Z. Ma, J. Ma, Y. Miao, X. Liu, K.-K. R. Choo, and R. Deng. Pocket diagnosis: Secure federated learning against poisoning attack in the cloud. *IEEE Trans. Serv. Comput.*

[28] M. Melis, A. Demontis, M. Pintor, A. Sotgiu, and B. Biggio. secml: A python library for secure and explainable machine learning. Dec. 2019. Accessed: Apr. 04, 2022.

[29] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava. Federated learning-based anomaly detection for iot security attacks. *IEEE Internet Things J.*, page 1, 2021.

[30] 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. *IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767, Jul. 2019.

[31] J. P. Nzabahimana. Analysis of security and privacy challenges in internet of things. *IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pages 175–178, May 2018.

[32] S. Otoum, N. Guizani, and H. Mouftah. Federated reinforcement learning-supported ids for iot-steered healthcare systems. *in ICC 2021 - IEEE International Conference on Communications*, pages 1–6, Jun. 2021.

[33] P. Papadopoulos, O. Thornewill von Essen, N. Pitropakis, C. Chrysoulas, A. Mylonas, and W. J. Buchanan. Launching adversarial attacks against network intrusion detection systems for iot. *J. Cybersecurity Priv.*, 1(2):252–273, Apr. 2021.

[34] A. Patcha and J. M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, 51(12):3448–3470, Aug. 2007.

[35] K. Pillutla, S. M. Kakade, and Z. Harchaoui. Robust aggregation for federated learning. Jan. 2022. Accessed: Mar. 17, 2022.

[36] R. Roman, J. Zhou, and J. Lopez. On the features and challenges of security and privacy in distributed internet of things. *Comput. Netw.*, 57(10):2266–2279, Jul. 2013.

[37] H. Saadat, A. Aboumadi, A. Mohamed, A. Erbad, and M. Guizani. Hierarchical federated learning for collaborative ids in iot applications. *in 2021 10th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–6, Jun. 2021.

[38] H. Sedjelmaci, S. M. Senouci, and M. Al-Bahri. A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology. *the IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[39] A. K. Singh, A. Blanco-Justicia, J. Domingo-Ferrer, D. Sanchez, and D. Rebollo-Monedero. Fair detection of poisoning attacks in federated learning. *in 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 224–229, Nov. 2020.

[40] Dr. S. Smys, Dr. Abul Basar, , and Dr. Haoxiang Wang. Hybrid intrusion detection system for internet of things (iot). *J. ISMAC*, 2(4):190–199, Sep. 2020.

[41] S. S. Swarna Sugi and S. R. Ratna. Investigation of machine learning techniques in intrusion detection system for iot network,. *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 1164–1167, Dec. 2020.

[42] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu. Data poisoning attacks on federated machine learning. *IEEE Internet Things J.*, page 1, 2021.

[43] I. Ullah and Q. H. Mahmoud. A two-level hybrid model for anomalous activity detection in iot networks. *16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6, Jan. 2019.

[44] J. W. Ulvila and J. E. Gaffney. Evaluation of intrusion detection systems. *J. Res. Natl. Inst. Stand. Technol.*, 108(6):453, Nov. 2003.

[45] A. Uprety and D. B. Rawat. Mitigating poisoning attack in federated learning. *in 2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, Dec. 2021.

[46] Q. Wang, R.-Q. Peng, J.-Q. Wang, Z. Li, and H.-B. Qu. Newlstm: An optimized long short-term memory language model for sequence prediction. *IEEE Access*, 8:65395–65401, 2020.

[47] C. Xie, K. Huang, P.-Y. Chen, and B. Li. Dba: Distributed backdoor attacks against federated learning. page 19, 2020.

[48] X. Xingmei, Z. Jing, and W. He. Research on the basic characteristics, the key technologies, the network architecture and security problems of the internet of things. *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, pages 825—-828, Oct. 2013.

[49] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):1–19, Feb. 2019.

[50] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, pages 1–7, Nov. 2015.

[51] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu. Poisongan: Generative poisoning attacks against federated learning in edge computing systems. *IEEE Internet Things J.*, pages 3310–3322, Mar. 2021.

[52] J. Zhang, G. Chunpeng, F. Hu, and B. Chen. Robustfl: Robust federated learning against poisoning attacks in industrial iot systems. *IEEE Trans. Ind. Inform.*, 2021.

VITA

Christian Dunham is Software Engineer for Nordstrom. He participates as a security advocate during the design review process. He welcomes your comments to `dunhamc@uw.edu`.