FedDef: Defense Against Gradient Leakage in Federated Learning-based Network Intrusion Detection Systems

Jiahui Chen, Yi Zhao, *Member, IEEE,* Qi Li, *Senior Member, IEEE,* Xuewei Feng, and Ke Xu, *Senior Member, IEEE*

***Abstract*—Deep learning (DL) methods have been widely applied to anomaly-based network intrusion detection system (NIDS) to detect malicious traffic. To expand the usage scenarios of DL-based methods, federated learning (FL) allows multiple users to train a global model on the basis of respecting individual data privacy. However, it has not yet been systematically evaluated how robust FL-based NIDSs are against existing privacy attacks under existing defenses. To address this issue, we propose two privacy evaluation metrics designed for FL-based NIDSs, including (1) privacy score that evaluates the similarity between the original and recovered traffic features using reconstruction attacks, and (2) evasion rate against NIDSs using adversarial attack with the recovered traffic. We conduct experiments to illustrate that existing defenses provide little protection and the corresponding adversarial traffic can even evade the SOTA NIDS Kitsune. To defend against such attacks and build a more robust FL-based NIDS, we further propose FedDef, a novel optimization-based input perturbation defense strategy with theoretical guarantee. It achieves both high utility by minimizing the gradient distance and strong privacy protection by maximizing the input distance. We experimentally evaluate four existing defenses on four datasets and show that our defense outperforms all the baselines in terms of privacy protection with up to 7 times higher privacy score, while maintaining model accuracy loss within 3% under optimal parameter combination.**

***Index Terms*—Federated Learning, Intrusion Detection, Gradient Privacy Leakage, Defense Strategy**

I. Introduction

DEEP learning-based network intrusion detection system (NIDS) has been widely used to detect malicious traffic and intrusion attacks, they are expected to raise alarms whenever the incoming traffic carries malicious properties (e.g., scan user's ports) or induces attacks (e.g., DDoS). Recently, researchers have been trying to adopt federated learning (FL), where multiple users collaboratively exchange information and train a global model with publicly shared gradients, to derive more accurate detection of cyberattacks without privacy leakage. For example, [1] proposes the collaboration-enabled intelligent Internet architecture for malicious traffic detection. [2] proposes a decentralized federated architecture that collaborates with eleven Internet Exchange Points (IXPs) operating in three different regions. It allows participant mitigation platforms to exchange information about ongoing amplification DDoS attacks. Experimental results illustrate that such collaboration can detect and mitigate up to 90% more DDoS attacks locally, which proves great success for faster and more effective detection and neutralization of attack traffic among collaborative networks. [3] also proposes an autonomous FL-based anomaly detection system to locate compromised Internet of Things (IoT) devices with high detection rate (95.6%) and fast inference ($257ms$), while [4] further proposes a FL-based NIDS for industrial cyber–physical systems in real world. These new researches indicate a promising trend that combination with FL can improve the overall detection performance of NIDS and is receiving extensive attention.

However, sharing model updates or gradients also makes FL vulnerable to inference attack in computer vision (CV) domain, e.g., property inference attack that infers sensitive properties of training data [5] and model inversion attack that reconstructs image training data [6]–[9]. Accordingly, there have also been some defense strategies such as differential privacy for deep learning [10] and Soteria [11] that perturbs the data representation features for images. However, unlike CV domain where slight noises added to the images can induce totally different visual perception and thus have higher tolerance for privacy, the reconstruction of traffic data may be more intimidating, because adversaries can evade the target model via (1) black-box attack that trains Generative Adversarial Network (GAN) model with the reconstructed benign data to generate new malicious traffic from random noises, or (2) white-box attack that directly perturbs the reconstructed malicious data. Unfortunately, few researches have systematically investigated to what extent current defenses combined with NIDS can protect user privacy, and whether there exists defense strategy that achieves both high utility and strong privacy guarantee to build a more robust FL-based NIDS.

Manuscript received 31 October 2022; revised 11 June 2023; accepted 3 July 2023. Date of publication x xxx xxxx; date of current version 14 July 2023. This work was supported in part by the National Science Foundation for Distinguished Young Scholars of China with No. 61825204, National Natural Science Foundation of China with No. 61932016, No. 62132011 and No. 62202258, Beijing Outstanding Young Scientist Program with No. BJJWZYJH012019100003011, China Postdoctoral Science Foundation with No. 2021M701894, China National Postdoctoral Program for Innovative Talents, and Shuimu Tsinghua Scholar Program. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. George Theodorakopoulos. *(Corresponding authors: Yi Zhao; Ke Xu.)*

Jiahui Chen, Yi Zhao, Xuewei Feng, and Ke Xu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: chenjiah22@mails.tsinghua.edu.cn; zhao_yi@tsinghua.edu.cn; fengxw06@126.com; xuke@tsinghua.edu.cn).

Qi Li is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China (e-mail: qli01@tsinghua.edu.cn).

Digital Object Identifier xxx/TIFS.xx.xx

To derive a more accurate evaluation of privacy for FL-based NIDSs, we propose two privacy metrics specifically designed for NIDS domain, i.e., privacy score and evasion rate. For the first one, we leverage reconstruction attacks, i.e., inversion and extraction attack, to recover the original training data from model gradients. Then we can calculate the similarity between raw and reconstructed data to evaluate privacy leakage, where we use different distance metrics for continuous (e.g., traffic duration) and discrete features (e.g., protocol type). With enough reconstructed traffic, we can evaluate evasion rate by training GAN model or applying perturbation to generate adversarial traffic to attack other NIDSs, which presents the practical threats in real world. With the above privacy metrics, we evaluate existing defenses combined with FL-based NIDS and demonstrate that all baselines fail to provide sufficient privacy protection and the adversarial traffic can even evade the SOTA NIDS Kitsune [12], which urges for new effective defense strategy.

To bridge this gap and build a more robust FL-based NIDS defending against gradient leakage, we further propose a novel input perturbation-based defense approach with theoretical guarantee for model convergence and data privacy named FedDef, which optimizes an objective function to transform the original input such that: 1) the distance between the new input and the raw input is as far as possible to prevent privacy leakage and 2) the corresponding gradients are as similar as possible to maintain model performance. Experimental results on four datasets illustrate that our defense can mitigate both reconstruction attacks and achieve at most 7 times higher privacy score compared to the second best defense, and the following adversarial attack fails to evade other NIDSs, which significantly outperforms other baselines. Regarding model performance, the FL model can still converge under our defense with iid and non-iid data distribution, and that model accuracy can be guaranteed within at most 3% loss with the optimal parameter combination.

Contributions. We summarize our contributions as follows:

- We propose two privacy evaluation metrics designed for FL-based NIDS, including (1) privacy score that evaluates the similarity between raw traffic feature and the recovered feature using reconstruction attacks, and (2) evasion rate against NIDSs using black-box and white-box adversarial attacks.
- To the best of our knowledge, we are the first to propose FedDef, an optimization-based input perturbation defense scheme for FL-based NIDS to prevent privacy leakage while maintaining model performance. To enhance our method, we also provide theoretical analysis for model convergence under non-iid data distribution and privacy guarantee for our defense.
- Experimental results on four datasets illustrate that our proposed FedDef outperforms existing defense approaches in terms of privacy protection. With FedDef, the privacy score is 1.5-7 times higher than the second best baseline during early training stage, and evasion rate on Kitsune model is always 0, while other baselines induce successful evasion more or less. In the meantime, FedDef still maintains high model utility with up to 3% accuracy loss compared to no-defense baseline under optimal parameter setting.

The rest of the paper is organized as follows: In Section II, we introduce some background and motivation. Section III provides our threat model and the corresponding privacy metrics with evaluation example. In Section IV, we introduce our defense design and detailed implementation with a representative FL framework. We provide theoretical analysis on convergence and privacy guarantee for our defense in Section V. In Section VI, we evaluate our defense with four baselines on four datasets with respect to privacy preserving and model performance. We discuss some future work in Section VII and conclude this study in Section VIII.

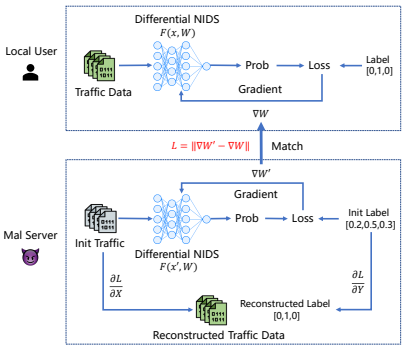


Fig. 1: Overview of inversion reconstruction attack.

II. BACKGROUND AND MOTIVATION

In this section, we introduce some background of FL-based NIDS, two SOTA reconstruction attacks and current defenses accordingly, and then we present the motivation of our study.

A. FL-based NIDSs

FL-based NIDS is a new promising topic that combines FL [13] like FedAvg [14] with intrusion detection. Local users first extract features from their private traffic data and then update the global model with the derived gradients, which are aggregated at the trusted server for later distribution. For example, [15] proposes a FL-based anomaly detection approach to proactively recognize intrusion in IoT networks using decentralized on-device data. Some work also consider hierarchical and interpretable FL to further build a practical NIDS [16], [17]. However, few have investigated the robustness of FL-based NIDSs against privacy attacks, which hinders the deployment of secure NIDSs.

B. Reconstruction Attack

FL also faces privacy leakage issues, such as data reconstruction [20], membership inference [25], and attribute inference [26]. We focus on the severest reconstruction attack that aims to recover training samples. It can be categorized into two types, i.e., optimization-based inversion attack and accurate extraction attack.

1) Inversion attack: DLG (i.e., Deep Leakage from Gradients) [20] solves an optimization problem to obtain the raw features and labels, which can be formulated as:

$$argmin_{x^*,y^*} \quad \|\nabla\theta(x^*, y^*) - \nabla\theta(x, y)\|_2 \tag{1}$$

TABLE I: A summary of existing defense approaches.

Approach	Category	Setting	Privacy	Utility	Theoretical Guarantee	Scalability	Feasibility
MPC [18]	MPC	●	●	●	●	◐	○
DP [10], [16], [19]	Gradient Perturbation	●	◐	◐	●	●	●
GP [20]	Gradient Perturbation	◐	◐	●	○	●	●
Lossless [21]	Gradient Perturbation	◐	●	●	●	◐	●
Instahide [22]	Input Perturbation	◐	◐	◐	○	●	◐
Soteria [11]	Input Perturbation	◐	○	●	●	◐	◐
ATS [23]	Input Perturbation	◐	◐	●	○	○	◐
DCS [24]	Input Perturbation	●	◐	●	○	◐	◐
FedDef (Ours)	Input Perturbation	●	●	●	●	●	●

Notes ●, ◐, ○ mean that the approach greatly, partly, barely considers setting metric or provides certain ability (other 5 metrics), respectively.

where $\nabla\theta(x,y) = \frac{\partial \ell(\theta,x,y)}{\partial \theta}$, ℓ is the local overall loss function on differentiable deep learning model (e.g., DNN), θ is the overall model parameter, and (x, y) is the original feature and label. Fig. 1 illustrates the process of inversion attack in FL, where malicious server (or user in decentralized mode) tries to match the publicly shared gradient with the dummy gradient. Specifically, they first randomly initialize a dummy input x^* and y^*. Then, they iteratively optimize the dummy gradients $\nabla\theta(x^*, y^*)$ close as to original by Euclidean distance, which also makes the dummy data close to the real training data.

Follow-up works improve on these results by using different distance metrics such as cosine similarity [7]:

$$argmin_{x^*,y^*} \quad 1 - \frac{\langle \nabla\theta(x^*, y^*), \nabla\theta(x, y)\rangle}{\|\nabla\theta(x^*, y^*)\|\|\nabla\theta(x, y)\|} \tag{2}$$

where $\langle\cdot,\cdot\rangle$ means inner product. [27] also proposes an improved DLG (i.e., iDLG) by analytically and accurately extracting ground-truth label from the last fully-connected (FC) layer, and thus it performs better as objective function (1) only has to optimize x^*.

2) Extraction attack: To further improve the reconstruction accuracy, researches [7], [28] propose an accurate extraction attack which can almost perfectly reconstruct a single training sample without any costs. This kind of attack assumes the model contains at least one layer, which consists of a weight and a bias. Specifically, we assume the first layer contains both weight and bias ([7] gives proof for more generalized scenario), then the direct output of the first layer y is computed as $W^T x + b$, where x is the raw input data and (W, b) is the corresponding weight and bias pair with compatible dimensionality. We further let i be the row of the first layer such that $\frac{\partial \ell}{\partial b_i} \neq 0$, then by chain rule we can obtain x as $x^T = (\frac{\partial \ell}{\partial b_i})^{-1} \frac{\partial \ell}{\partial W_i^T}$. In this way, we can extract the exact data as long as there exists $\frac{\partial \ell}{\partial b_i} \neq 0$. In Section III, we present a specific solution of x^* using PyTorch and give proof for its failure when batch size of input data x is more than 1.

C. Defenses

There exist several defenses to mitigate such reconstruction attacks in FL and can be categorized into three types: secure multi-party computation (MPC), gradient perturbation, and input perturbation. Existing representative defense approaches are summarized in TABLE I in terms of 6 metrics, where setting describes iid or non-iid training mode, utility evaluates the model performance, scalability denotes whether such defense can be applied in different domains, and feasibility denotes whether it requires extra setup or data to perform such defense.

1) Secure multi-party computation: MPC [18], [29] allows a group of parties to synchronously compute a function and obtain accurate representations of the final value while protecting their private data from privacy leakage. For example, [30] proposes to let users encrypt their local updates such that the central server can only recover the aggregation of the updates. However, MPC requires special setup and can be costly to implement, and [5] shows that adversaries can still launch inference attacks against MPC for certain scenarios.

2) Gradients perturbation: These defenses modify the gradients before updating them to the server. For example, [20] proposes gradient pruning (GP) that sets certain percent of the derived gradients of small absolute magnitudes to zero, which serves as a gradient mask to hide data information. Differential privacy (DP) [19] provides theoretical privacy guarantee and bounds the change in output distribution caused by a small input noise like Gaussian noise [10], [16]. [21] also proposes to add and eliminate random noises to the global model from the server to prevent information leakage from malicious clients, yet it works only when the server is trusted.

3) Input perturbation: These defenses tend to perturb or mix the original data from the source. For example, [22] proposes Instahide to encode the training data with user's private dataset instead of blind noises. [11] proposes Soteria to perturb the image feature representation before the defended layer to confuse the reconstruction attack. In the meantime, [23] proposes to search for optimal image transformation combination such as image rotation and shift to preserve privacy. However, there is no one-size-fits-all traffic transformation pattern and may not apply to NIDS domain. [24] proposes to obfuscate the gradients of sensitive data with concealing data to preserve privacy, yet it limits the percentage of sensitive data within a batch and does not consider label protection.

D. Motivation: Why privacy protection for FL-NIDS

In FL, if the training data are images or contexts that carry high-level sensitive information, the adversary can directly reconstruct the data and manipulate such privacy. While for

FL-NIDS, the training data are tabular features with low-level information, which can be less straightforward to understand. However, traffic data reconstruction can also pose great threats because those features contain unique patterns tailored to the clients, which can be manipulated for adversarial attacks.

1) Privacy in traffic: To reveal the necessity for privacy protection, we first present some important features of traffic data in intrusion detection scenario.

• **Static info** includes source/destination IP and port, network protocol, service, and specific flags.

• **Timestamp** represents the arrival time of a packet. Aggregation of enough packets can carry information about traffic behavior pattern. For example, TANTRA [31] utilizes DNN to learn benign traffic's timestamp differences and apply them on malicious data for adversarial attacks.

• **Packet size** represents the content length of a packet. Traffic flows usually follow specific packet size distribution under certain network protocol. For instance, [32] leverages this feature to generate blind adversarial perturbations to evade DNN-based traffic classifiers.

• **Flow throughput** represents the packet sending or receiving rate during a traffic flow. Adversary can construct malicious traffic with moderate flow rate that mimics the target traffic tailored to the target model like DDoS attack in [33].

• **Others** include the combination of basic features like max/min packet size during a TCP flow between specific IPs. These features are usually perturbed to help maximize the probability of a successful evasion.

2) Why privacy protection: In FL-NIDS, as introduced in Section II-B, the adversary can recover raw traffic data via reconstruction attacks. Once he/she obtains enough samples with the features mentioned above, instead of targeting the data themselves like images, the adversary can make use of the data to perform the following actions:

• Learning benign traffic's behavior to generate malicious traffic with black-box attacks, e.g., training GAN with benign data and generate with random noises, or just learning the timestamp feature for more subtle attacks as TANTRA [31].

• Directly perturbing the recovered malicious data with white-box attacks targeting specific model, e.g., PGD [34], DeepFool [35], and AutoPGD [36].

Note that we will conduct extensive experiments to validate the efficiency of such black-box and white-box attacks in Section VI-C, as a straightforward demonstration for potential threats of reconstruction attacks even with existing defenses, and therefore the necessity for stronger privacy protection for FL-NIDS. As a result, in this paper we are motivated to propose a new effective defense strategy FedDef to enhance the data security for FL in Section IV.

III. Threat Model and Privacy Metric

In this section, we present our adversary and make several assumptions, then we introduce two privacy metrics with an evaluation example to enhance our motivation.

A. Adversary

We consider an honest-but-curious server that follows the exact FL protocol and is allowed to observe the updates from different users. The goal of the attacker is to reconstruct each user's private data and launch adversarial attack against NIDSs. We make several assumptions of the adversary's power as follows: (1) The adversary is aware of the model architecture and loss function. (2) The adversary knows about the property of the training data, including max/min values, and feature types (i.e., discrete or continuous). (3) The adversary is aware of each user's local training batch size. In this way, the adversary can perform suitable attacks depending on different scenarios and thus evaluate defenses' lower bound.

B. Privacy Score via Reconstruction Attack

1) Inversion Attack in NIDS scenario: We leverage optimization-based reconstruction attack using L_2 and cosine distance metrics as introduced in Section II-B. However, there exist two challenges when it comes to traffic features:

Firstly, traffic features are usually normalized during preprocessing procedure, therefore, the reconstructed data x^* should satisfy $x^* \in [0,1]^{dim}$, where dim is the dimension of data features. Secondly, there are discrete and continuous traffic features in x^*, which require different techniques.

To address the first challenge, we can project the final optimized x^* into legal feature space such that $x^* = clamp(x^*, min = 0, max = 1)$. Note that we don't leverage variable change since it induces additional computation and achieves similar performance. To address the second challenge, we first project the normalized x^* into original feature space, and then we force the discrete features to be integer and normalize x^* again for later privacy evaluation.

Note that when the batch size of the training data is more than 1, inversion attack may not converge because the reconstructed data can have different permutations according to different initialization. Therefore, we reconstruct a single sample at a time while keeping the rest the same. We incorporate label y^* in the optimization process and output the index of the maximum value of y^* as the reconstructed label.

2) Extraction attack in NIDS scenario: In Section II-B, we have illustrated that extraction attack is effective for a single training sample whenever there is a layer with both weight and bias. According to the neural network in PyTorch, we have the following theorem:

Theorem 1: If the first layer has a bias, then $x^T = (\frac{\partial \ell}{\partial W})^T(\frac{\partial \ell}{\partial b})^T(\frac{\partial \ell}{\partial b}(\frac{\partial \ell}{\partial b})^T)^{-1}$ when and only when batch size=1.

Proof of Theorem 1: When batch size is 1, the output y of a FC layer is computed as $y = xW^T + b$, where x is the input with size $1 \times in\_feature$, (W, b) are the weight and bias with size $out\_feature \times in\_feature$ and $1 \times out\_feature$, respectively, and we have $(\frac{\partial \ell}{\partial W})^T = \frac{\partial \ell}{\partial W^T} = x^T\frac{\partial \ell}{\partial y} = x^T\frac{\partial \ell}{\partial b}$, then we have:

$$x^T = (\frac{\partial \ell}{\partial W})^T(\frac{\partial \ell}{\partial b})^T(\frac{\partial \ell}{\partial b}(\frac{\partial \ell}{\partial b})^T)^{-1} \tag{3}$$

When batch size is more than 1, since b is broadcast into $b^* = [b, ..., b]$ with size $batch \times out\_feature$, we have $\frac{\partial \ell}{\partial b^*}(\frac{\partial \ell}{\partial b^*})^T = \begin{bmatrix} bb^T & ... & bb^T \\ ... & bb^T & ... \\ bb^T & ... & bb^T \end{bmatrix} = bb^T \begin{bmatrix} 1 & ... & 1 \\ ... & 1 & ... \\ 1 & ... & 1 \end{bmatrix}$,

where the second matrix ($batch \times batch$) is not invertible and Eq. (3) fails. ■

Definition 1 (Privacy Score): Based on the reconstructed data, we introduce privacy score metric where we directly add up the absolute distance for continuous features, and then we project the data into their original feature space and set the distance of discrete features to 1 if they don't match the original value. The overall score computation is :

$$score(x, x^*) = \frac{\sum_{i \in S_c} |x_i - x_i^*| + \sum_{i \in S_d} equal(X_i, X_i^*)}{|S_c| + |S_d|} \tag{4}$$

where S_c and S_d denote the continuous and discrete features, X and X^* are projected features with original value ranges, $equal(x_1, x_2) = 1$ if $x_1 = x_2$, and it gets 0 otherwise. In this way, we can quantitatively evaluate the privacy leakage for each defense for FL-based NIDS.

C. Evasion Rate via Adversarial Attack

As long as the adversary obtains enough reconstructed data using either inversion attack or extraction attack, he/she can launch black-box attack by filtering out benign traffic to train GAN to generate adversarial examples from random noises. In addition, the adversary can also perform white-box attack like PGD [34] that directly perturbs the reconstructed malicious traffic to reach better evasion rate than black-box methods.

Definition 2 (Evasion Rate): We derive evasion rate (ER) using the generated adversarial traffic to attack the trained DNN-based FL model and Kitsune. For more straight evasion results, we present model accuracy ACC_{DNN} for DNN model where $ER = 1 - ACC_{DNN}$, and Root Mean Squared Error $RMSE$ for Kitsune model, where $ER = 1$ when $RMSE$ is lower than threshold and $ER = 0$ otherwise.

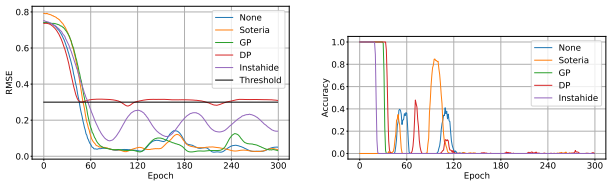
D. Evaluation Example

1) Example of privacy score: To better understand the privacy metrics, we first evaluate model utility and privacy score on several existing defenses (optimal parameters) against inversion attack and list some reconstructed normalized features in TABLE II, where x and x^* denote the original and reconstructed traffic feature, respectively. Smaller score generally means more privacy leakage. Model without defense is the most vulnerable with the lowest privacy score (6.6e-4) and the adversary can almost perfectly recover continuous and discrete traffic features, while the other four defenses can achieve higher score (3.3e-3 to 2.8e-1) with less data leakage. However, deploying defenses can degrade the FL model utility, where Instahide performs the worst with 4.6% accuracy loss in exchange for such privacy guarantee.

2) Example of evasion rate: To further demonstrate the consequences of reconstruction attack, we leverage black-box attack as an example and evaluate evasion rate of adversarial examples (AEs) on Kitsune [12] and trained DNN-based FL-NIDS (refer to Section VI-C for setup). Fig. 2 illustrates the RMSE and accuracy change during the training process. We can find that even DP with such strong privacy guarantee can sometimes induce successful evasion. For example, RMSE for DP is under the threshold for KDD99 dataset against Kitsune and thus $ER = 1$, while for the DNN model, the accuracy also drops to 0 for Mirai dataset. The other defenses can only perform worse, which poses great potential threats to users. We also notice that AEs under DP perform relatively worse than other baselines, which corresponds to TABLE II that DP achieves higher privacy score (2.8e-1) and induces less data privacy leakage.

TABLE II: Reconstruction example on KDD99 dataset, **ACC** denotes the FL model accuracy, **PS** denotes privacy score. We present normalized and original value for discrete features*.

Data \ Feature	ACC	PS	Duration	Protocol*	Service*	Src_bytes	Dst_bytes
x	\	\	0.00	1.00/2	0.16/12	2.01e-4	0.00
x^* w/o. defense	0.996	6.6e-4	0.00	1.00/2	0.16/12	4.60e-4	0.00
x^* w/. Soteria [11]	0.994	3.3e-3	0.00	1.00/2	0.16/12	0.00	1.38e-3
x^* w/. GP [20]	0.989	1.5e-1	0.11	0.50/1	0.16/12	0.00	0.00
x^* w/. DP [10]	0.985	2.8e-1	0.10	1.00/2	0.00/1	0.00	0.00
x^* w/. Instahide [22]	0.950	2.0e-1	0.00	0.50/1	0.37/26	0.00	0.00



(a) KDD99+Kitsune (b) Mirai+DNN

Fig. 2: Evasion rate on Kitsune and FL-NIDS models for KDD99 and Mirai dataset.

In summary, our proposed metrics are practical to evaluate the robustness of FL-based NIDSs combined with different defenses, and the example results demonstrate the insufficiency of existing defenses. Therefore, it is critical to design a new defense approach to build a more robust NIDS that preserves traffic data privacy without damaging model performance.

IV. FedDef: Optimization-based Input Perturbation Defense

In this section, we present our defense FedDef. We introduce detailed optimization design and also implementation with a representative FL framework, i.e., FedAvg.

A. Overview of FedDef

As illustrated in Fig. 3, local users first download the latest global model from the server. During local training, users first leverage FedDef to transform their own private data into pseudo data such that the data are dissimilar to preserve privacy and the corresponding gradients are similar to preserve FL model performance. Then the users update the corresponding pseudo gradient instead of the original gradient.

The right side in Fig. 3 illustrates the adversary manipulating the reconstructed data to attack the target models. The adversary first leverages reconstruction attack to recover user's private training data and labels from the gradients, then black-box adversarial attack is available using GAN to evade the FL model or other NIDSs such as Kitsune. While our defense ensures that the adversary can reconstruct only the pseudo data, which cannot help GAN model converge and therefore the adversarial attack fails.

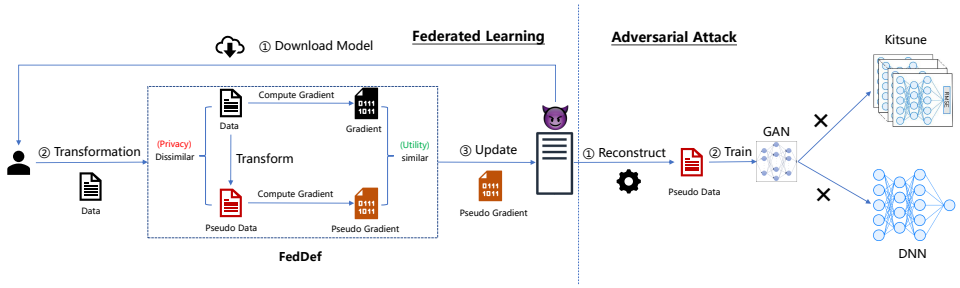


Fig. 3: Overview of FedDef.

B. Optimization Design of FedDef

Privacy preserving and model performance are two key indicators for FL, users collaboratively train a global robust model to acquire high detection results on intrusion attacks while also protecting their private data from information leakage. Based on these two aspects, we design a novel defense scheme for NIDS, i.e., FedDef, aiming to (1) preserve data privacy against reconstruction attack and the following adversarial attack, and (2) maintain FL model performance.

1) Privacy preserving: To address the privacy concern, we prevent privacy leakage from the source. The input training data should be dissimilar from the original data, and we have the following optimization:

$$argmax_{x',y'} \quad D(x', y'; x, y) \tag{5}$$

where x and y are the original input data and labels, x' and y' are the pseudo pair for later pseudo gradient computation. $D(\cdot,\cdot)$ denotes the distance metric that evaluates how dissimilar (x', y') is from (x, y), where features and labels require different techniques.

Data Transformation. According to the inversion attack and extraction attack introduced in Eq. (1) and Eq. (3), the reconstructed features x^* depend on the absolute numerical values of x', therefore, we optimize x' to be as far from x as possible by L_2 distance metrics as $argmax_{x'} \quad \|x' - x\|_2$.

However, the reconstructed data x^* is always projected into $[0, 1]^{dim}$, where dim is the feature dimension of x and x^*. Therefore, it makes no difference to the reconstructed data x^* when $x^* \leq 0$ or $x^* \geq 1$. Due to the consideration above, we can constrain the upper bound of the L_2 distance and rewrite the optimization with respect to x' as

$$argmin_{x'} \quad ReLU(\delta - \|x' - x\|_2) \tag{6}$$

where δ is the upper bound of the distance between x and x', and $ReLU(x) = max(0, x)$, which ensures the distance is as large as possible while also guaranteeing the upper bound.

Label Transformation. On the contrary, the reconstructed labels y^* do not rely on the absolute numerical values of y', instead the adversary takes the index i of the maximum label probability y_i^* as the final label, therefore increasing the overall distance between y' and y does not guarantee label privacy. For example, let i be the index of y, $y_i = 1$ and $y_j = 0, \forall j \in [1, n], j \neq i$, where n is the total label classes. By leveraging L_2 distance metric to optimize y', we may obtain y' where $y'_i = 2$ and $y'_j = 1, \forall j \in [1, n], j \neq i$. In this way, though the overall distance between y' and y is quite large, the adversary may still acquire the correct label i since y'_i is the maximum one in y' and the reconstructed label y_i^* may also share the same property, therefore, label information is extracted by equation $i = argmax(y^*)$.

To address this problem, we choose to minimize y'_i so that it stays the minimum one in y' and the adversary will probably get any indexes other than the ground-truth label i. Specifically, we first locate the label index i in y, and we minimize y' such that the minimum one y'_{min} is as close to y'_i as possible which can be formulated as $argmin_{y'} \quad |y'_{min} - y'_i|$, where $|\cdot|$ denotes the absolute distance. In this way, we optimize y'_i to be the minimum one in y' so that the adversary will most probably reconstruct any indexes except i.

Finally, we formulate the optimization for privacy preserving as follows:

$$argmin_{x',y'} \quad ReLU(\delta - \|x' - x\|_2) + |y'_{min} - y'_i| \tag{7}$$

where i denotes the ground-truth label index such that $y_i = 1$, and y'_{min} is the minimum one in y'.

2) Model performance: To maintain FL model performance, we choose to add a constraint for the pseudo gradient $\nabla\theta(x', y')$ generated by the pseudo data and label (x', y') to be as close to the original gradient $\nabla\theta(x, y)$ as possible. In particular, we expect the L_2 distance between the pseudo gradient and the original gradient to be within ϵ boundary, which can be written as

$$s.t. \ \|\nabla\theta(x', y') - \nabla\theta(x, y)\|_2 \leq \epsilon \tag{8}$$

However, such constraint is highly non-linear and thus we transform the constraint into an optimization problem using $ReLU$ function as

$$argmin_{x',y'} \quad ReLU(\|\nabla\theta(x', y') - \nabla\theta(x, y)\|_2 - \epsilon) \tag{9}$$

Finally, we can combine the two objective functions to optimize (x', y') such that the pseudo data and the original data are dissimilar to preserve data privacy and the corresponding gradients are similar to maintain model performance. In particular, we can derive the combined optimization as follows:

$$\begin{aligned} &argmin_{x',y'} \quad L_{Pri} + \alpha L_{Per} \\ &L_{Pri} = ReLU(\delta - \|x' - x\|_2) + |y'_{min} - y'_i| \\ &L_{Per} = ReLU(\|\nabla\theta(x', y') - \nabla\theta(x, y)\|_2 - \epsilon) \end{aligned} \tag{10}$$

where L_{Pri} and L_{Per} stand for optimization loss for privacy preserving and FL model performance, respectively. α is a parameter that evaluates the tradeoff between privacy and model performance. The overall algorithm is illustrated in **Algorithm 1**.

Algorithm 1: Transformation of private data and label

Input: original data pair (x, y), global differentiable model parameters θ, loss function F, defense epochs def_ep, privacy tradeoff α, learning rate def_lr, constant g_value, ϵ, δ

Output: pseudo data pair (x', y')

1 $x' \leftarrow U(0,1), y' \leftarrow U(0,1)$; ▷ random initialization
2 $\nabla\theta(x,y) \leftarrow \frac{\partial F(\theta,x,y)}{\partial\theta}$; ▷ original gradient
3 $gt \leftarrow argmax(y)$; ▷ ground-truth label index of y
4 **for** $i \leftarrow 1$ *to* def_ep **do**
5 $\quad\nabla\theta(x',y') \leftarrow \frac{\partial F(\theta,x',y')}{\partial\theta}$; ▷ pseudo gradient
6 $\quad$**if** $max(|\nabla\theta(x',y')|) \leq g\_value$ **then return** x', y';
7 $\quad\quad$ ▷ early stop
8 $\quad loss \leftarrow ReLU(\|\nabla\theta(x',y') - \nabla\theta(x,y)\|_2 - \epsilon)$ ▷ L_2 distance
9 $\quad y'_{min} \leftarrow min(y')$;
10 $\quad loss \leftarrow \alpha * loss + ReLU(\delta - \|x' - x\|_2) + |y'_{min} - y'_{gt}|$;
11 $\quad x' \leftarrow Adam(x', \frac{\partial loss}{\partial x'}, def\_lr)$; ▷ Adam optimizer
12 $\quad y' \leftarrow Adam(y', \frac{\partial loss}{\partial y'}, def\_lr)$;
13 **end**
14 **return** x', y'

Note that we leverage early stop to terminate the optimization as long as the pseudo gradient tends to vanish. In particular, we first set a predetermined constant g_value, then we check the maximum absolute value of all gradients of the FL model and stop the optimization whenever the maximum value is lower than g_value (line 6). In this way, we prevent extreme case of vanishing gradient and find it effective against extraction attack, because the pseudo gradients are small enough that $\frac{\partial F}{\partial b}(\frac{\partial F}{\partial b})^T$ approaches 0 due to calculation accuracy and thus matrix inversion in Eq. (3) fails, and global model can still get updated in the meantime.

C. Implementation With FedAvg

We introduce how to combine FedDef with the classic FL framework, i.e., FedAvg [14], to build a robust FL-based NIDS. We first formulate our FL training objective in FedAvg:

$$argmin_\theta \ \{F(\theta) = \sum_{k=1}^{N} p_k F_k(\theta)\} \tag{11}$$

where N is the total number of local users, p_k is the corresponding weight of the k-th user participating in the global model training, and $p_k \geq 0$, $\sum_{k=1}^{N} p_k = 1$. θ is the parameter of the global differentiable model, $F_k(\cdot)$ is the overall loss function for the k-th user (e.g., cross entropy loss). N local users collaboratively train a global NIDS model by solving optimization (11) iteratively. Here we present one training round (e.g., t-th) with our defense as follows:

(1) The server has learnt a global model θ_t at t-th round and distributes the model to all the participants.

(2) Every local user (e.g., k-th) first initializes their model with θ_t, i.e., $\theta_t^k = \theta_t$ and then performs $E \geq 1$ local updates. Specifically, during each update, user first leverages FedDef to transform their private data and label with the latest model θ_t^k to generate pseudo data pair (x', y') for pseudo gradient computation. Therefore, the local model can be updated for the i-th iteration as:

$$\xi'^k_{t+i} = FedDef(\xi^k_{t+i}) \tag{12}$$

$$\theta^k_{t+i+1} \leftarrow \theta^k_{t+i} - \eta_{t+i}\nabla F_k(\theta^k_{t+i}, \xi'^k_{t+i}) \ , i = 0, 1, ..., E-1 \tag{13}$$

where ξ^k_{t+i} is a sample pair (i.e., (x^k_{t+i}, y^k_{t+i})) uniformly chosen from the k-th user's private dataset. $FedDef$ is our defense algorithm that outputs the pseudo data pair ξ'^k_{t+i}, η_{t+i} is the learning rate for the i-th iteration. In this way, user k performs Eq. (12) and Eq. (13) for E rounds and finally obtains the local trained model θ^k_{t+E}.

Note that at each training round, the pseudo data pair ξ'^k_{t+i} can be different due to different input data pair ξ^k_{t+i} and the newly updated FL model.

(3) After local training is complete, the central server determines a set $\mathcal{S}_t$, which contains a subset of K indices randomly selected with replacement according to the sampling probabilities $p1, ..., p_N$ from the N users. Then the global model is aggregated by simply averaging as:

$$\theta_{t+E} \leftarrow \frac{1}{K}\sum_{k\in\mathcal{S}_t} \theta^k_{t+E} \tag{14}$$

In this way, the global model θ_{t+E} can be updated with only partial users, which can mitigate serious "straggler's effect", where the server has to wait for the slowest (even offline) user's update and faces great time delay.

To evaluate the model performance of such FL-NIDS, we directly feed the testing dataset to the DNN model without $FedDef$ transformation to derive clean accuracy.

V. Theoretical Analysis

In this section, we derive theoretical analysis of model convergence and privacy guarantee for FedDef.

A. Convergence Analysis

We provide theoretical analysis for FL model convergence using FedAvg combined with our defense. Our analysis follows [37] on the convergence of FedAvg on non-iid data with partial users' update.

We first make the five following assumptions same as [37]:

Assumption 1: $F_1, ..., F_N$ are all L-smooth, that is, for all V and W, and any $k \in [1, N]$, $F_k(V) \leq F_k(W) + (V - W)^T\nabla F_k(W) + \frac{L}{2}\|V - W\|_2^2$.

Assumption 2: $F_1, ..., F_N$ are all μ-strongly convex, that is, for all V and W, and any $k \in [1, N]$, $F_k(V) \geq F_k(W) + (V - W)^T\nabla F_k(W) + \frac{\mu}{2}\|V - W\|_2^2$.

Assumption 3: Let ξ_t^k be sampled from the k-th user's local data uniformly at random. The variance of stochastic gradients for each user is bounded: $\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^k) - \nabla F_k(\theta_t^k)\|^2 \leq \sigma_k^2$, for all $k = 1, ..., N$.

Assumption 4: The expected squared norm of stochastic gradients is uniformly bounded: $\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^k)\|^2 \leq G^2$, for all $k = 1, ..., N$ and $t = 0, ..., T-1$, T is the total iterations of local users' updates.

***Assumption** 5:* Assume $\mathcal{S}_t$ contains a subset of K indices randomly selected with replacement according to the sampling probabilities $p1, ..., p_N$. The aggregation step of FedAvg performs $\theta_t \leftarrow \frac{1}{K}\sum_{k\in\mathcal{S}_t}\theta_t^k$.

***Theorem** 2:* Let F^* and F_k^* be the minimum values of F and F_k, respectively. We use the term $\Gamma = F^* - \sum_{k=1}^N p_k F_k^*$ to quantify the degree of non-iid. Recall that E denotes the local updates for each user and T denotes the total iterations. Let Assumptions 1-5 hold and L, μ, σ_k, G be defined therein. Choose $\kappa = \frac{L}{\mu}$, $\gamma = max\{8\kappa, E\}$, and learning rate $\eta_t = \frac{2}{\mu(\gamma+t)}$. Then FedAvg with our defense with partial users participation for non-iid data satisfies:

$$\mathbb{E}[F(\theta_T)] - F^* \leq \frac{2\kappa}{\mu+T}(\frac{B+C}{\mu} + 2L\|\theta_0 - \theta^*\|^2) \quad (15)$$

where

$$\begin{aligned} B &= \sum_{k=1}^N p_k^2(\epsilon+\sigma)_k^2 + 6L\Gamma + 8(E-1)^2(\epsilon+G)^2 \\ C &= \frac{4}{K}E^2(\epsilon+G)^2 \end{aligned} \quad (16)$$

***Proof of Theorem 2**:* We replace Assumptions 3-4 with our pseudo gradients and then derive the Theorem 2. Without the loss of generality, we consider the k-th user's pseudo gradients' property. We first present the following lemmas:

***Lemma** 1:* Let $\|\cdot\|_2$ be a sub-multiplicative norm, for all A and B, we have $\|A\|_2 - \|B\|_2 \leq \|A+B\|_2 \leq \|A\|_2 + \|B\|_2$ and $\|AB\|_2 \leq \|A\|_2\|B\|_2$.

***Lemma** 2:* Let $\|\cdot\|_2$ be a sub-multiplicative norm, for all A and B, we have $\mathbb{E}^2\|A\|_2\|B\|_2 \leq \mathbb{E}\|A\|_2^2\mathbb{E}\|B\|_2^2$.

The proof of Lemma 1 and 2 can be naturally obtained with norm triangle inequality and Cauchy–Schwarz inequality.

***Lemma** 3:* Let $\|\cdot\|_2$ be a sub-multiplicative norm, for all A and B, we have $\mathbb{E}\|A+B\|_2^2 \leq (\sqrt{\mathbb{E}\|A\|_2^2} + \sqrt{\mathbb{E}\|B\|_2^2})^2$.

***Proof of Lemma 3**:*

$$\begin{aligned} \mathbb{E}\|A+B\|_2^2 &\leq \mathbb{E}(\|A\|_2 + \|B\|_2)^2 \\ &= \mathbb{E}\|A\|_2^2 + \mathbb{E}\|B\|_2^2 + 2\mathbb{E}\|A\|_2\|B\|_2 \\ &\leq \mathbb{E}\|A\|_2^2 + \mathbb{E}\|B\|_2^2 + 2\sqrt{\mathbb{E}\|A\|_2^2\mathbb{E}\|B\|_2^2} \\ &= (\sqrt{\mathbb{E}\|A\|_2^2} + \sqrt{\mathbb{E}\|B\|_2^2})^2 \end{aligned} \quad (17)$$

■

Replace Assumption 3. According to our optimization (10), the pseudo gradients' L_2 distance from the original are constrained within ϵ. Therefore, the following equation holds:

$$\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^{'k}) - \nabla F_k(\theta_t^k, \xi_t^k)\|_2^2 \leq \epsilon^2 \quad (18)$$

The variance of the pseudo stochastic gradients for each user is bounded with Lemma 3:

$$\begin{aligned} &\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^{'k}) - \nabla F_k(\theta_t^k)\|^2 \\ =&\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^{'k}) - \nabla F_k(\theta_t^k, \xi_t^k) + \nabla F_k(\theta_t^k, \xi_t^k) - \nabla F_k(\theta_t^k)\|_2^2 \\ \leq&(\sqrt{\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^{'k}) - \nabla F_k(\theta_t^k, \xi_t^k)\|_2^2} \\ &+ \sqrt{\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^k) - \nabla F_k(\theta_t^k)\|_2^2})^2 \\ \leq&(\epsilon + \sigma_k)^2 \end{aligned} \quad (19)$$

Replace Assumption 4. We leverage similar method to replace Assumption 4 with our pseudo gradient, the expected squared norm of pseudo stochastic gradients is uniformly bounded:

$$\begin{aligned} &\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^{'k})\|^2 \\ =&\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^{'k}) - \nabla F_k(\theta_t^k, \xi_t^k) + \nabla F_k(\theta_t^k, \xi_t^k)\|_2^2 \\ \leq&(\sqrt{\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^{'k}) - \nabla F_k(\theta_t^k, \xi_t^k)\|_2^2} + \sqrt{\mathbb{E}\|\nabla F_k(\theta_t^k, \xi_t^k)\|_2^2})^2 \\ \leq&(\epsilon + G)^2 \end{aligned} \quad (20)$$

In this way, we can replace the bound of the original gradients in Assumptions 3-4 with our pseudo gradients. Therefore, Assumptions 1-5 hold with our defense. By applying our new bounds in [37], we can derive Theorem 2 aforementioned.

■

B. Privacy Analysis

We provide theoretical analysis for data privacy with our defense. Specifically, we derive a possible lower bound for the deviation of pseudo data x' from the original x using the derivatives of the first layer. We first make the following assumption:

***Assumption** 6:* Assume that there exist both weight W and bias b in the first layer of the model, and that the norm of the gradient with respect to the bias is bounded for some $M > 0$: $\|\frac{\partial F}{\partial b}\|_2 \leq M$.

***Theorem** 3:* Let Assumption 6 hold, (x, y) be the original data pair after normalization, (x', y') be the transformed data pair with our defense, $\nabla b = \frac{\partial F(\theta,x,y)}{\partial b}$, $\nabla W = \frac{\partial F(\theta,x,y)}{\partial W}$,$\nabla b' = \frac{\partial F(\theta,x',y')}{\partial b}$, $\nabla W' = \frac{\partial F(\theta,x',y')}{\partial W}$ and we have the following theorem to guarantee data privacy.

$$\|x' - x\|_2 \geq \frac{2(\|\nabla W' - \nabla W\|_2 - \|\nabla b' - \nabla b\|_2)}{2M + \|\nabla b' - \nabla b\|_2} \quad (21)$$

***Proof of Theorem 3**:* We can derive that $\nabla W^T = x^T\nabla b$ and $\nabla W'^T = x'^T\nabla b'$, therefore, we have the following transformation:

$$2(\nabla W - \nabla W')^T = (x-x')^T(\nabla b + \nabla b') + (x+x')^T(\nabla b - \nabla b') \quad (22)$$

With Lemma 1, we can derive the inequality on the one hand:

$$\begin{aligned} \|(x-x')^T(\nabla b + \nabla b')\|_2 \leq&\|x-x'\|_2\|\nabla b + \nabla b'\|_2 \\ \leq&\|x-x'\|_2(\|\nabla b\|_2 + \|\nabla b'\|_2) \\ \leq&2M\|x-x'\|_2 \end{aligned} \quad (23)$$

Note that $\|x\|_2 \leq 1$ since x is normalized. Therefore, on the other hand, we also have:

$$\begin{aligned} &\|(x-x')^T(\nabla b + \nabla b')\|_2 \\ =&\|2(\nabla W - \nabla W')^T - (x+x')^T(\nabla b - \nabla b')^T\|_2 \\ \geq&2\|(\nabla W - \nabla W')\|_2 - \|(x+x')^T(\nabla b - \nabla b')^T\|_2 \\ \geq&2\|(\nabla W - \nabla W')\|_2 - \|(x+x')\|_2\|(\nabla b - \nabla b')\|_2 \\ =&2\|(\nabla W - \nabla W')\|_2 - \|(x'-x+2x)\|_2\|(\nabla b - \nabla b')\|_2 \\ \geq&2\|(\nabla W - \nabla W')\|_2 - (\|x'-x\|_2 + 2\|x\|_2)\|(\nabla b - \nabla b')\|_2 \\ \geq&2\|(\nabla W - \nabla W')\|_2 - (\|x'-x\|_2 + 2)\|(\nabla b - \nabla b')\|_2 \end{aligned} \quad (24)$$

Therefore, by combining the above inequalities, we have :

$$2M\|x'-x\|_2 \geq 2\|(\nabla W-\nabla W')\|_2-(\|x'-x\|_2+2)\|(\nabla b-\nabla b')\|_2 \tag{25}$$

By extracting $\|x'-x\|_2$, we have Theorem 3. ■

VI. Evaluation

In this section, we conduct several experiments to evaluate our defense from two perspectives: (1) Model utility under iid and non-iid data distribution. (2) Privacy protection using the two proposed metrics, i.e., privacy score (plus label reconstruction accuracy) and evasion rate. We also conduct ablation study to derive optimal parameters combination and test computation overhead.

A. Experimental Setup

We conduct our experiments using PyTorch with 8-core, 64GB CPU and one Nvidia-P100 (16GB) GPU. The basic experiments setting is as follows:

1) Datasets: We choose four network intrusion attack datasets to evaluate our defense, including KDDCUP'99 [38], Mirai botnet [12], CIC-IDS2017 [39], and UNSW-NB15 [40]. Mirai botnet contains only one type of malicious traffic named Botnet Malware and the other three datasets contain multiple attack types. To further balance the datasets, we select only DDoS attack traffic from CIC-IDS2017 so that we evaluate on two 2-class dataset (i.e., Mirai Botnet and CIC-IDS2017) and two multi-class datasets (i.e., KDDCUP'99 and UNSW-NB15). More details can be found in TABLE III.

TABLE III: Datasets used in our work

Property / Dataset	Feature Dimension	Attack Type	Training Samples	Testing Samples
KDD99	41	23	345815	148206
Mirai	100	2	210000	90000
CIC-IDS2017	76	2	157998	67713
UNSW-NB15	42	10	180372	77301

2) FL model: We derive our global NIDS model under FedAvg framework with 10 local users collaboratively training a 3-layer fully connected DNN with layer sizes [dim, $2*dim$, $3*dim$, n], where dim denotes the feature size and n denotes attack types. The first two layers both consist of a linear network with ReLU activation function, while the third layer also has a linear network with softmax that projects the output vector into label class probability.

3) Baselines and parameters configuration: We choose five baselines mentioned in Section II-C to compare with our work, including no defense, Soteria, gradient pruning (GP), differential privacy (DP), and Instahide.

For our defense, we set the default parameter as $\epsilon = 0$, $\delta = 1$, $g\_value = 1e-15$, learning rate $def\_lr = 2e-1$, and steps $def\_ep = 40$ unless otherwise specified, we also set $\alpha = 0.25, 0.5, 1$ accordingly to observe its impact.

For other defenses, we optimize the parameters to achieve the best tradeoff between performance and privacy. Specifically, we set the gradient compression rate to 99% for GP, and we leverage Laplace noise and set the mean and variance of the noise distribution as 0 and 0.1 for DP, while following the default setting for Soteria and Instahide.

4) Evaluation metrics: We leverage FL model accuracy to evaluate defense's utility. For privacy guarantee, we derive privacy score (plus label reconstruction accuracy) and evasion rate as introduced in Section III.

B. Model Performance Results

Setup 1. We evaluate FL model accuracy as introduced in Section VI-A with Adam optimizer with learning decay. We first divide the original datasets to derive training dataset and testing dataset by $7:3$, and each user can get certain amount of the training dataset under iid or non-iid setting.

We design a new non-iid data generation algorithm. Specifically, each user can randomly get the same percent of the benign traffic, and then we randomly select $p > 0$ attack types and distribute the same percent ($\frac{1}{10}$ in our case) of that specific malicious traffic. To balance the result, we consider non-iid data distribution for multi-class KDD99 dataset, and iid data distribution for the other three datasets, where each user gets the same percent of the whole training dataset. After dataset distribution, local users normalize their own datasets with maximum/minimum values.

We set the overall training rounds $T = 300$, and each user updates the global model only once $local\_ep = 1$ with batch size $local\_bs = 1000$. We follow the same parameter setting introduced in Section VI-A and we optimize the local training learning rate for different defenses and datasets. Detailed parameters are illustrated in TABLE IV, where no defense is around 1e-2, ours is around 1.5e-2, and strong defense like DP requires higher learning rate as 3e-2, otherwise such great noises to the gradients can hardly help optimize the model. Note that we also leverage learning rate decay to accelerate the model convergence. Specifically, the learning rate lr is updated every 20 rounds as $lr = lr \times decay$, therefore, lr is getting smaller as the model is approaching convergence to avoid performance oscillation.

Accuracy Analysis. We run the training for five times and report the best results for baselines and the average accuracy for our defense in TABLE V. It can be found that our defense with $\alpha = 1$ achieves similar performance from the baseline with at most 2.6% accuracy loss (from UNSW dataset) and the least deviation under iid or non-iid setting. As α tends to get smaller, Eq. (10) tends to optimize pseudo data's privacy more than pseudo gradients' utility, which is why FedDef with $\alpha = 0.25$ achieves lower accuracy with greater deviation (0.01 at most). Meanwhile, we also compare our defense with other baselines and find that Soteria achieves the highest accuracy with only 1% loss at most while Instahide generally performs the worst. We suspect that the reason for Instahide's poor performance is that it randomly flips certain signs of the mixed data to provide additional security protection in exchange of model performance.

In summary, the FL model can still converge and reach high accuracy combined with our defense, thanks to our constraints on gradient deviation.

C. Model Privacy Results

We evaluate model privacy with privacy score and evasion rate for different local training batches and training stages.

TABLE IV: Parameter setting for model performance evaluation presented as learning rate/decay.

Dataset \ Defense	No Defense	Our Defense			Soteria	GP	DP	Instahide
		$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.25$				
KDD99	1e-2/0.9	1.5e-2/0.9	1.5e-2/0.9	1.5e-2/0.9	1e-2/0.9	3e-2/0.9	3e-2/0.9	3e-2/0.9
Mirai	3e-3/0.9	3e-3/0.9	3e-3/0.9	3e-3/0.9	3e-3/0.9	3e-3/0.9	3e-2/0.9	3e-3/0.9
CIC-IDS2017	1e-3/0.8	1.5e-2/0.8	1.5e-2/0.8	1.5e-2/0.8	1e-3/0.8	3e-3/0.9	3e-2/0.9	3e-3/0.9
UNSW	3e-3/0.8	1.5e-2/0.9	1.5e-2/0.9	1.5e-2/0.9	2e-3/0.9	3e-2/0.9	3e-2/0.9	3e-2/0.8

TABLE V: Accuracy comparison on four datasets, higher is better.

Dataset \ Defense	No Defense	Our Defense			Soteria	GP	DP	Instahide
		$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.25$				
KDD99	0.996	0.987±0.001	0.986±0.001	0.985±0.002	0.994	0.989	0.985	0.950
Mirai	0.923	0.922±0.000	0.922±0.000	0.921±0.000	0.923	0.922	0.922	0.920
CIC-IDS2017	0.982	0.971±0.002	0.970±0.004	0.968±0.005	0.978	0.970	0.971	0.964
UNSW-NB15	0.746	0.720±0.008	0.710±0.009	0.706±0.010	0.743	0.710	0.705	0.689

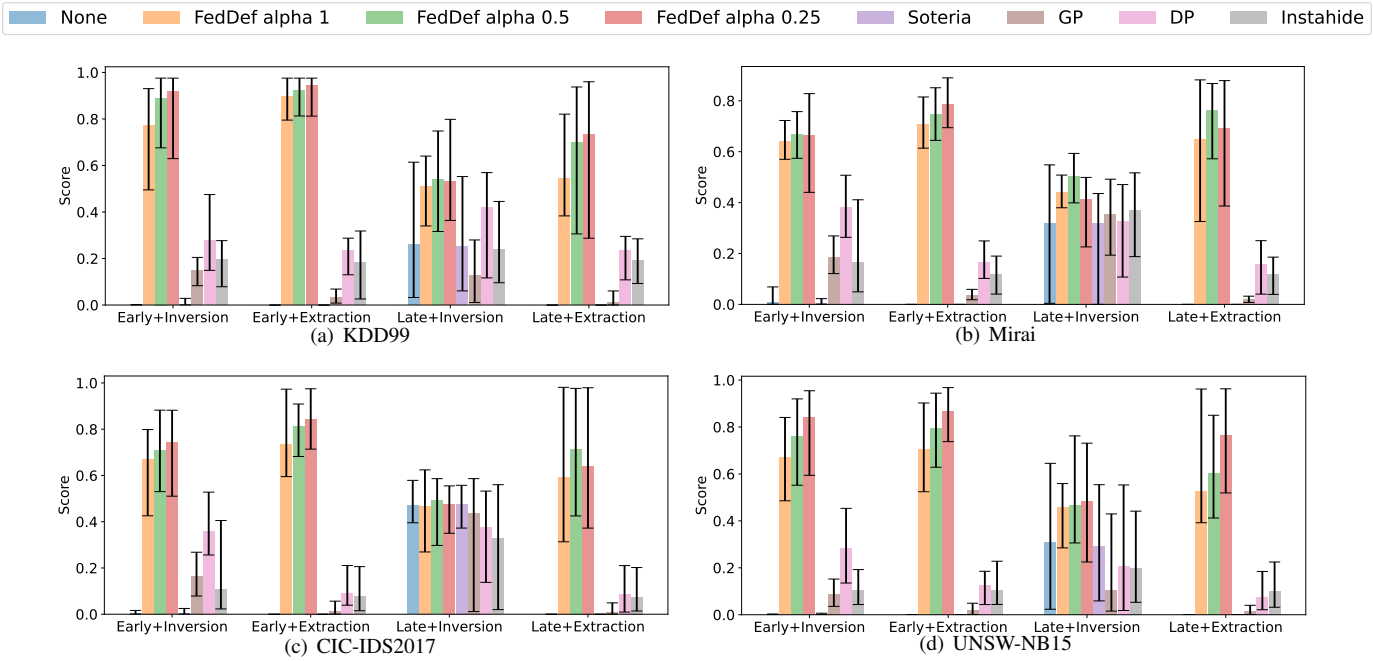


Fig. 4: Reconstruction privacy score comparison on KDD99 and Mirai datasets, higher is better.

1) Single sample reconstruction: We first consider local users updating model only once with only one training sample. In this scenario, the adversary can launch both inversion attack and extraction attack.

Setup 2. We evaluate privacy score with inversion attack using appropriate distance metric to optimize the reconstruction, i.e., we use cosine distance for KDD99 and CIC-IDS2017 datasets during late training stage and L_2 for the rest of the cases. Note that extraction attack may fail due to calculation accuracy when gradients are small, and we will fall back to inversion attack when Eq. (3) fails. As introduced in Section II-B, we can only derive data from extraction attack, therefore, we leverage optimization-based inversion attack to acquire labels. We follow the same parameter setting as in Section VI-A, and we set the overall iteration $T = 100$, local batch size $local\_bs = 1$, local step $local\_ep = 1$ for both early and late stages. Note that we use the randomly initialized model for early stage and the respective trained model in Section VI-B for late training stage, we do not actually update the model to constantly evaluate privacy score.

TABLE VI: Label reconstruction accuracy comparison on four datasets, and lower is better.

Data	Stage \ Defense	No Defense	Our $\alpha = 1$	Our $\alpha = 0.5$	Our $\alpha = 0.25$	Soteria	GP	DP	Instahide
KDD99	Early	1.00	0.01	0.00	0.00	1.00	1.00	0.99	0.37
	Late	0.59	0.17	0.17	0.16	0.75	0.99	0.29	0.22
Mirai	Early	1.00	0.97	0.86	0.81	1.00	1.00	1.00	0.52
	Late	0.53	0.50	0.49	0.45	0.58	0.80	0.68	0.48
CIC-IDS2017	Early	1.00	0.92	0.91	0.74	1.00	1.00	1.00	0.53
	Late	0.52	0.48	0.45	0.44	0.55	0.45	0.53	0.47
UNSW-NB15	Early	1.00	0.06	0.01	0.00	1.00	1.00	1.00	0.26
	Late	0.71	0.18	0.06	0.06	0.81	0.98	0.77	0.22

Privacy Score Analysis. The full results for average privacy score and reconstructed label accuracy over the T samples are in Fig. 4 and TABLE VI. During early training stage, the reconstruction attack proves excellent performance, where the privacy score can reach almost 0 without any defense and labels are also accurately extracted (ACC=1) among four datasets. The baseline defenses provide some but limited

protection, especially against more accurate extraction attack.

On the contrary, our defense outperforms other baselines and significantly mitigates such attacks in a way that the score is around 0.6-0.7 with $\alpha = 1$, which is 1.5-7 times higher than the second best DP with privacy score around 0.1-0.4, and it tends to get even higher with smaller α and against stronger extraction attack. Our defense also prevents the label leakage especially for multi-class datasets with almost 0 label inversion accuracy compared to 0.26-1.00 for baselines, while it may perform worse for two-class datasets where labels are easier to obtain. Overall, the privacy is well protected with little information leakage. Interestingly, Soteria almost proves no privacy protection against either attack, we suspect it's because we don't leverage CNN as our feature extractor and that Soteria only perturbs the gradients of the defended layer while the rest still carry much private information.

During the late training stage, the shared gradients tend to get smaller, and such optimization-based attack performs worse because it's more difficult to optimize the dummy gradient to fit the original one. As illustrated in Fig. 4 and TABLE VI, the privacy score using inversion attack varies from 0.2 to 0.5 even for gradients without defenses, and the label accuracy also drops from 1 to even 0.52 for CIC-IDS2017 dataset. Nonetheless, our defense still generally outperforms other baselines where extraction attack performs with similar privacy score to that of early stage because matrix inversion is always accurate. We also notice that smaller α may induce even lower score because the gradients are of small magnitude and thus may trigger falling back from extraction to inversion attack, which makes the reconstruction results more unstable.

Setup 3. We first evaluate evasion rate (ER) using black-box attack as introduced in Section III-C. For early stage, we recover 100 reconstructed benign traffic using extraction attack to train GAN to generate 100 randomly initialized adversarial examples (AEs) against two NIDSs, i.e., Kitsune and the trained global DNN model with corresponding defenses. While we repeat the process for late stage with inversion attack for our defense with $\alpha = 1$ only since extraction attack presents similar privacy score. For Kitsune setup, we train it with the four datasets and determine optimal thresholds respectively. For the trained DNN-based FL model, we only consider either normal or malicious for evasion results.

Black-box Adversarial Attack. The evasion results for early and late stages are in Fig. 5 and Fig. 6. For early stage, we can find that almost all baselines fail to prevent such adversarial attack ($ER = 1$ for RMSEs lower than threshold) except for Mirai dataset, where threshold is more strict and DP may be sufficient with strong privacy guarantee, while our defense consistently outperforms baselines that the curve under FedDef doesn't converge with higher RMSE around 0.8-1.5, which is 2-15 times higher than threshold and thus AEs fail to evade Kitsune ($ER = 0$). However, it's easier to evade the target DNN model ($ER = 1$ when $ACC_{DNN} = 0$) even with our defense for CIC-IDS2017 dataset. It's because we also leverage DNN model to train GAN, therefore, evading discriminator also means likely evasion on DNN model.

While in late training stage, inversion attack can be quite unstable thus the recovered data approaches random guess, which is why RMSE is similar to that of randomly initialized data (RMSE is still higher than threshold and $ER = 0$), and accuracy (also for ER) is around 0.5.

2) Batched samples reconstruction: We study a more practical scenario with multi-sample reconstruction.

Setup 4. We evaluate ER using inversion attack to reconstruct data and labels because we proved in Section III that extraction attack is only effective when batch size is 1. Note that we don't consider the specific privacy score because the reconstructed data may have different permutations and it's hard to correspond them to the ground-truth data and label. Instead, we directly apply the data with benign labels to train the GAN model for black-box attack scenario.

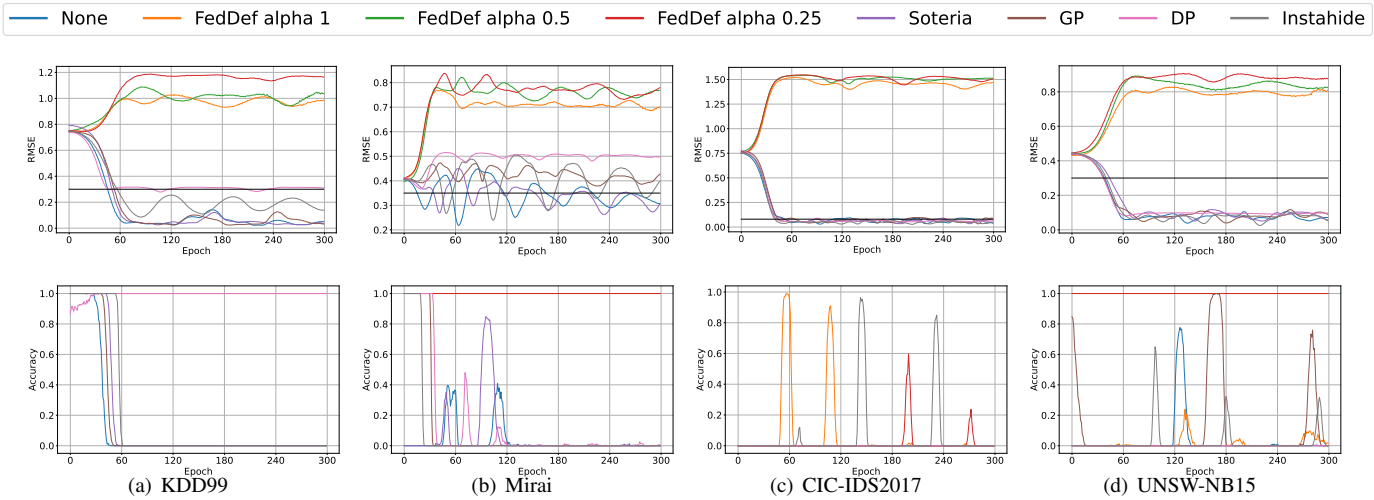
Black-box Adversarial Attack. The representative results for KDD99 dataset with batch size 5 and 10 are illustrated in Fig. 7. We can observe that with larger batch size, the inversion attack may perform worse and therefore may slightly degrade our defense (RMSE is around 0.8-1.0 compared to 1.0-1.2 when batch size=1), yet the overall results are similar to that of single sample reconstruction ($ER = 0$ for FedDef and $ER = 1$ for baselines mostly), which further demonstrates the threats in practical training.

*3) **White-box adversarial attack**:* To further evaluate the privacy leakage in reconstructed data, we also conduct several white-box adversarial attacks against the two NIDSs.

Setup 5. Recall that for GAN-based black-box attack, we recover benign traffic so as to generate malicious traffic from random examples. While in white-box attack scenario, we directly reconstruct malicious training data and perform different adversarial attacks on the samples themselves to evade the target model.

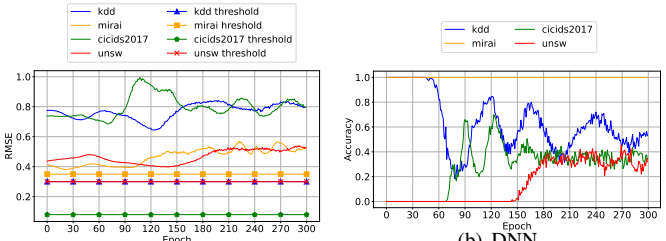
Specifically, we recover 100 malicious traffic (based on the reconstructed label) using inversion attack per sample in early stage for different defenses. Then we perform the following attacks to generate adversarial examples:

- FGSM [41] utilizes the sign of the gradient of the cross-entropy (CE) loss associated with the target label to obtain perturbation.
- CW [42] generates adversarial perturbations by solving a norm-restricted constrained optimization problem, where the loss has in contrast to the CE loss a direct interpretation in terms of the decision of the classifier.
- PGD [34] is an extension of the FGSM. It initializes the attack at a random point in the L_p ball constraint and projects the perturbation back onto the L_p ball after every iteration.
- DeepFool [35] is an untargeted attack under the assumption that deep learning models are linear, with a decision boundary (i.e., hyperplanes) separating each class. On every iteration, DeepFool linearizes the classifier around the current point x and computes the perturbations as an orthogonal projection vector that projects x onto the closest hyperplane.
- AutoPGD [36] is a budget-aware step size-free variant of PGD that induces failures due to suboptimal step size and the objective function. Instead it automatically adjusts the step size and chooses Difference of Logits Ratio (DLR) loss, which is both shift and rescaling invariant, and thus has the same degrees of freedom as the decision of the classifier.



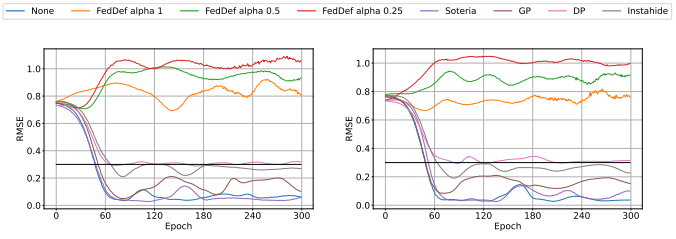
(a) KDD99 (b) Mirai (c) CIC-IDS2017 (d) UNSW-NB15

Fig. 5: Black-box adversarial attack against two NIDSs with extraction attack for single sample reconstruction during early stage. The first row is the average RMSE score change (higher is better) for Kitsune during the GAN training process, the black line represents the optimal threshold, and the second row represents the DNN accuracy change (lower is better).



(a) Kitsune (b) DNN

Fig. 6: Black-box adversarial attack against two NIDSs with inversion attack for single sample against our defense with $\alpha = 1$ during late stage.



(a) Batch Size=5 (b) Batch Size=10

Fig. 7: Black-box adversarial attack for KDD99 dataset with inversion attack against Kitsune during early stage when batch size$= 5/10$.

For implementaion, we leverage the open-source package torchattacks [43], which provides unified API for different adversarial attacks. For attack initialization, we present some key parameters for each attack in TABLE VII. Note that for CW attack, we leverage parameter c to adjust the weights for distance optimization. For DeepFool, the perturbation is usually too large to be valid, therefore, we manually add ϵ parameter to constrain the maximum perturbation on the final output to ensure practical attack. For other parameters, we follow the default setting in torchattacks for each attack.

For the trained DNN model, we apply the 5 attacks mentioned above in targeted mode by labels where malicious traffic are projected into benign class, except for untargeted DeepFool that perturbs the traffic in the direction of the closest label class. Specifically, we empirically set $\epsilon = 40/255, \alpha = 6/255$ for KDD99, Mirai, UNSW datasets, $\epsilon = 4/255, \alpha = 2/255$ for CIC2017 dataset, and $c = 0.01, step = 100$ for all scenarios.

TABLE VII: White-box attack parameter setting.

Attack	Key Parameter	Description
FGSM	ϵ	maximum perturbation
CW	c	constraint for L_2 distance
PGD	ϵ, α, $step$	maximum perturbation, single perturbation, attack step size
DeepFool	ϵ	maximum perturbation
AutoPGD	ϵ	maximum perturbation

For the unsupervised Kitsune model, the output is the anomaly score instead of possibility vector. Therefore, DeepFool and AutoPGD are unavailable because DeepFool requires the label classification of the adversarial examples to compute the perturbation towards the closest class. While AutoPGD also needs the output of DNN vector to obtain DLR loss and update the step size. Therefore, we only adapt FGSM, CW, and PGD attacks, where we change the CE loss to the anomaly score output from Kitsune. In this way, we can minimize the score to fool the model to classify the adversarial examples as benign as long as the score is lower than the threshold.

In this scenario, we can conduct ablation study on the key parameters since anomaly score can convey direct impact of the parameter on the adversarial attacks. Specifically, we choose $c = 1e-2, 1e-3, 1e-4$ for CW attack and $\epsilon = 10/255, 40/255, 80/255$ for PGD attack, $\epsilon = 40/255$ is constant for FGSM attack since it's a simple version of PGD.

DNN Analysis. TABLE VIII illustrate the results of white-box adversarial attacks against DNN model. We present the classification accuracy and the perturbation distance of the adversarial examples under different attacks and defenses. Higher accuracy means better defense because the reconstructed traffic are further from the original distribution of the user training data, therefore, limited perturbation is not enough to evade the

TABLE VIII: White-box adversarial attack against DNN model. We present as (model accuracy)/(L_2 perturbation distance) as below, higher is better from defenders' view.

Datasets	Attack \ Defense	No Defense	FedDef			Soteria	GP	DP	Instanhide
			$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.25$				
KDD99	FGSM	6%/0.12	62%/0.12	100%/0.13	100%/0.12	25%/0.11	24%/0.13	9%/0.14	52%/0.13
	CW	24%/0.10	63%/0.08	100%/0.00	100%/0.00	42%/0.08	0%/0.09	1%/0.09	50%/0.06
	PGD	0%/0.11	57%/0.11	100%/0.12	100%/0.09	22%/0.11	24%/0.12	6%/0.13	47%/0.12
	DeepFool	7%/0.11	64%/0.10	100%/0.02	100%/0.02	27%/0.11	28%/0.10	23%/0.06	92%/0.01
	AutoPGD	1%/0.12	61%/0.11	100%/0.08	100%/0.09	22%/0.10	24%/0.13	7%/0.14	81%/0.09
Mirai	FGSM	0%/0.14	75%/0.11	100%/0.12	100%/0.11	27%/0.14	64%/0.13	56%/0.13	50%/0.14
	CW	31%/0.03	91%/0.01	100%/0.00	100%/0.00	67%/0.04	0%/0.17	2%/0.16	12%/0.11
	PGD	0%/0.13	75%/0.11	100%/0.12	100%/0.11	24%/0.14	64%/0.13	47%/0.12	44%/0.13
	DeepFool	0%/0.12	75%/0.11	100%/0.12	100%/0.11	28%/0.13	90%/0.11	66%/0.11	95%/0.06
	AutoPGD	0%/0.13	75%/0.08	100%/0.07	100%/0.09	24%/0.13	63%/0.10	47%/0.11	65%/0.11
CIC-IDS2017	FGSM	0%/0.01	74%/0.01	100%/0.01	100%/0.01	0%/0.01	0%/0.01	2%/0.01	20%/0.02
	CW	0%/0.00	69%/0.01	100%/0.00	100%/0.00	0%/0.00	0%/0.01	0%/0.01	0%/0.03
	PGD	0%/0.01	74%/0.01	100%/0.01	100%/0.01	0%/0.01	0%/0.01	2%/0.01	20%/0.01
	DeepFool	0%/0.00	74%/0.01	100%/0.00	100%/0.00	0%/0.00	4%/0.00	2%/0.00	33%/0.00
	AutoPGD	0%/0.00	74%/0.01	100%/0.01	100%/0.01	0%/0.00	0%/0.01	2%/0.01	30%/0.01
UNSW-NB15	FGSM	18%/0.12	71%/0.13	91%/0.12	100%/0.10	10%/0.12	0%/0.13	3%/0.13	2%/0.14
	CW	1%/0.09	75%/0.02	100%/0.05	100%/0.00	0%/0.08	0%/0.03	1%/0.05	0%/0.04
	PGD	0%/0.12	65%/0.13	91%/0.11	100%/0.09	0%/0.12	0%/0.12	0%/0.13	2%/0.13
	DeepFool	46%/0.07	90%/0.03	100%/0.03	100%/0.02	44%/0.07	32%/0.01	37%/0.02	22%/0.00
	AutoPGD	0%/0.12	79%/0.10	91%/0.10	100%/0.10	0%/0.12	0%/0.13	1%/0.14	11%/0.12

target model. The evasion rate is consistent with the definition in Section III-C that $ER = 1 - ACC_{DNN}$.

From defenders' view, we can conclude that our defense FedDef achieves the highest accuracy for most of the cases when $\alpha = 1$ (57%-91%) and all the cases when $\alpha = 0.5, 0.25$ (91%-100%). This corresponds with the fact that lower α means better privacy optimization, therefore, the reconstructed malicious traffic carry less information and can be harder to be perturbed. While Instahide also performs well with high accuracy, which means that mixed training data can already achieve moderate privacy guarantee by incorporating different label classes. Soteria and GP still perform worse since the reconstructed data are just similar to the baseline defense.

From attackers' view, PGD attack generally works better with lower accuracy (also higher evasion rate) with reasonable perturbation, followed by FGSM, CW, AutoPGD, and DeepFool. This illustrates that DLR loss in AutoPGD may not be an optimal choice compared with traditional CE loss in this case. While DeepFool only targets the nearest label class instead of benign class and thus cannot work as well as other targeted attacks.

In some scenarios, CW attack works better than PGD with higher accuracy and lower perturbation range (e.g., 12% vs. 44% for Instahide defense for Mirai dataset), thanks to the optimization of both perturbation distance and evasion rate. During the experiments, some adversarial examples are not perturbed at all (distance=0), this is because CW only updates the examples when classification reaches the target benign label and the perturbation distance is lower. Therefore, it won't perform perturbation if the reconstructed malicious data themselves are benign (0%/0.00 for No defense for CIC2017 dataset) since distance=0 is already the lowest case, or if the adversarial examples can never reach benign class (100%/0.00 for FedDef when $\alpha = 0.5$ for Mirai and CIC2017 datasets).

Kitsune Analysis. We present model accuracy and the average anomaly score for Kitsune model in TABLE IX. Among the 100 perturbed reconstructed traffic data, those with scores lower than the model threshold are considered to successfully evade Kitsune. Therefore, higher anomaly score and accuracy means better defense performance.

For better comparison, we also present the anomaly score for each defense before the adversarial attack in TABLE X. From defenders' view, FedDef still outperforms all the baselines in terms of both accuracy and score. We can find that the anomaly score for FedDef before the attack is already high enough (0.77-1.24 when $\alpha = 1$) and thus model accuracy is always 100%. In contrast, for KDD99 and UNSW datasets, the original scores for no defense (0.27/0.12), Soteria (0.27/0.11) and GP (0.25/0.08) are low enough to evade Kitsune, let alone the adversarial score. While DP and Instahide present relatively higher scores, yet adversarial perturbation can still render some scores lower than the threshold, resulting in 16%-81% and 85%-100% accuracy for KDD99 respectively. This corresponds with the privacy score in Fig. 4. In other words, FedDef outperforms DP and Instahide, which again outperform GP and Soteria.

However, the threshold for CIC2017 is so low (0.08). Even PGD attack can hardly render the adversarial examples successful. PGD with $\epsilon = \frac{80}{255}$ perturbation range can only achieve 91% for no defense. Yet we can still come to similar conclusion as DNN model analysis for other three datasets for each defense.

From attackers' view, PGD attack still works better than FGSM and CW. Specifically, given the same $\epsilon = \frac{40}{255}$, PGD can achieve lower score and accuracy (e.g., 4%/0.31 and 8%/0.32 for Mirai dataset for no defense). While CW with lower distance constraint c can also reach better evasion rate, PGD generally works better if we correspond c and ϵ in order (e.g., 0%/0.16-25%/0.25 and 0%/0.24-25%/0.27 for KDD99 for no defense).

Note that CW attack also requires successful evasion (DNN's target classification or scores lower than the threshold)

TABLE IX: White-box adversarial attack against Kitsune model. We present as (model accuracy)/(anomaly score) as below, higher is better from defenders' view.

Datasets	Attack \ Defense		No Defense	FedDef			Soteria	GP	DP	Instanhide
				$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.25$				
KDD99	FGSM		19%/0.21	100%/0.88	100%/0.97	100%/0.94	17%/0.21	0%/0.17	44%/0.31	97%/0.61
	CW	$c = 1e-2$	25%/0.27	100%/1.06	100%/1.16	100%/1.13	27%/0.26	24%/0.24	81%/0.42	100%/0.76
		$c = 1e-3$	19%/0.26	100%/1.06	100%/1.15	100%/1.13	8%/0.25	6%/0.23	42%/0.38	95%/0.75
		$c = 1e-4$	0%/0.24	100%/1.06	100%/1.15	100%/1.13	0%/0.25	4%/0.23	19%/0.34	85%/0.73
	PGD	$\epsilon = 10/255$	25%/0.25	100%/1.01	100%/1.11	100%/1.08	26%/0.25	24%/0.22	81%/0.39	100%/0.74
		$\epsilon = 40/255$	16%/0.21	100%/0.87	100%/0.96	100%/0.94	14%/0.21	0%/0.16	42%/0.30	97%/0.61
		$\epsilon = 80/255$	0%/0.16	100%/0.70	100%/0.77	100%/0.75	0%/0.16	0%/0.10	16%/0.21	85%/0.48
Mirai	FGSM		8%/0.32	100%/0.60	100%/0.65	100%/0.63	1%/0.31	47%/0.34	89%/0.41	79%/0.41
	CW	$c = 1e-2$	67%/0.39	100%/0.77	100%/0.82	100%/0.81	86%/0.40	97%/0.47	100%/0.57	100%/0.57
		$c = 1e-3$	29%/0.36	100%/0.77	100%/0.82	100%/0.81	68%/0.39	59%/0.43	100%/0.57	98%/0.57
		$c = 1e-4$	17%/0.36	100%/0.77	100%/0.82	100%/0.81	34%/0.37	18%/0.36	75%/0.52	25%/0.42
	PGD	$\epsilon = 10/255$	90%/0.38	100%/0.72	100%/0.78	100%/0.76	93%/0.37	93%/0.43	100%/0.52	100%/0.53
		$\epsilon = 40/255$	4%/0.31	100%/0.60	100%/0.65	100%/0.63	0%/0.31	44%/0.33	84%/0.41	75%/0.41
		$\epsilon = 80/255$	0%/0.23	100%/0.44	100%/0.48	100%/0.46	0%/0.23	0%/0.22	9%/0.28	14%/0.27
CIC-IDS2017	FGSM		100%/0.19	100%/1.03	100%/1.06	100%/1.14	100%/0.19	100%/0.26	100%/0.60	100%/0.58
	CW	$c = 1e-2$	100%/0.26	100%/1.24	100%/1.28	100%/1.40	100%/0.28	100%/0.41	100%/0.77	100%/0.77
		$c = 1e-3$	100%/0.26	100%/1.24	100%/1.28	100%/1.37	100%/0.28	100%/0.41	100%/0.77	100%/0.77
		$c = 1e-4$	78%/0.23	100%/1.24	100%/1.28	100%/1.37	82%/0.26	49%/0.27	100%/0.77	100%/0.77
	PGD	$\epsilon = 10/255$	100%/0.23	100%/1.19	100%/1.22	100%/1.31	100%/0.25	100%/0.36	100%/0.72	100%/0.72
		$\epsilon = 40/255$	100%/0.17	100%/1.03	100%/1.05	100%/1.14	100%/0.18	100%/0.25	100%/0.59	100%/0.57
		$\epsilon = 80/255$	91%/0.12	100%/0.82	100%/0.84	100%/0.91	90%/0.11	78%/0.14	100%/0.44	100%/0.40
UNSW-NB15	FGSM		0%/0.08	100%/0.64	100%/0.70	100%/0.73	0%/0.08	0%/0.07	10%/0.22	40%/0.27
	CW	$c = 1e-2$	0%/0.12	100%/0.80	100%/0.87	100%/0.90	0%/0.11	0%/0.08	49%/0.32	82%/0.40
		$c = 1e-3$	0%/0.12	100%/0.80	100%/0.87	100%/0.90	0%/0.11	0%/0.08	12%/0.29	33%/0.34
		$c = 1e-4$	0%/0.12	100%/0.80	100%/0.87	100%/0.90	0%/0.11	0%/0.08	2%/0.28	10%/0.30
	PGD	$\epsilon = 10/255$	0%/0.10	100%/0.76	100%/0.82	100%/0.86	0%/0.09	0%/0.05	48%/0.29	82%/0.36
		$\epsilon = 40/255$	0%/0.06	100%/0.64	100%/0.70	100%/0.73	0%/0.06	0%/0.02	8%/0.21	36%/0.27
		$\epsilon = 80/255$	0%/0.03	100%/0.49	100%/0.54	100%/0.57	0%/0.03	0%/0.02	1%/0.13	1%/0.17

TABLE X: Original anomaly score of the reconstructed malicious data against Kitsune model.

Dataset \ Defense	Threshold	No Defense	FedDef			Soteria	GP	DP	Instahide
			$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.25$				
KDD99	0.30	0.27	1.06	1.16	1.13	0.27	0.25	0.43	0.76
Mirai	0.35	0.41	0.77	0.83	0.81	0.40	0.47	0.57	0.57
CIC-IDS2017	0.08	0.26	1.24	1.28	1.37	0.28	0.41	0.77	0.77
UNSW-NB15	0.30	0.12	0.80	0.87	0.90	0.11	0.08	0.32	0.40

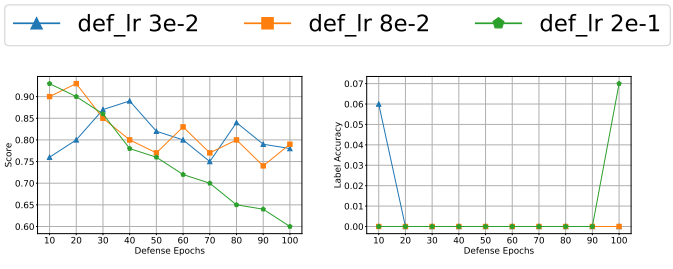
to actually update the adversarial example, therefore, it also cannot perturb FedDef's reconstructed data due to our better privacy protection (the same anomaly score as the original). While PGD attack can always set higher ϵ to ensure successful evasion (at higher perturbation price), which is more feasible than CW since constraint c has a limit 0.

4) Conclusion: We briefly conclude our privacy analysis. In general, our defense outperforms all current defenses with high privacy score and low evasion rate even with strong model performance guarantee ($\epsilon = 0$) for single or multiple samples, in both training stages, against either privacy attack, and the following black-box and white-box adversarial attacks.

D. Ablation Study

So far, we have demonstrated our defense's model performance and privacy preserving guarantee. Next we will study some parameter impact on the overall performance of FedDef.

Setup 6. We evaluate model accuracy and privacy score (plus label accuracy) on KDD99 dataset to study the optimal parameter combination with respect to α, def_lr and def_ep with determined ϵ and δ. Firstly, we choose our optimal $\alpha = 1$ because it induces the best accuracy performance and privacy score is high enough to mitigate the adversarial attack. We follow the same experiment setting for model performance and privacy evaluation as introduced in Section VI-B and VI-C.



(a) Early+Inversion (b) Early+Label Accuracy

Fig. 8: Ablation study on def_lr during early training stage on KDD99 dataset with inversion attack for single sample reconstruction.

Ablation Analysis. The average privacy score and accuracy results over five training times can be found in TABLE XI and Fig. 8. We can find that higher learning rate generally reduces the steps needed to fully optimize pseudo data. Specifically, it takes $def\_lr = 3e-2, 8e-2, 2e-1$ about 70, 50, 40 steps accordingly to achieve model accuracy as high as 0.987, while also achieving similar privacy score as high as 0.78.

TABLE XI: Accuracy comparison on KDD99 datasets for different parameters with baseline 0.996, higher is better.

def_lr \ def_ep	10	20	30	40	50	60	70	80	90	100
3e-2	0.442±0.186	0.534±0.087	0.917±0.040	0.965±0.015	0.977±0.003	0.983±0.002	0.985±0.002	0.987±0.002	0.987±0.001	0.987±0.001
8e-2	0.497±0.193	0.968±0.003	0.980±0.003	0.986±0.002	0.987±0.002	0.989±0.001	0.989±0.001	0.988±0.001	0.990±0.002	0.989±0.001
2e-1	0.967±0.006	0.984±0.002	0.984±0.002	0.987±0.001	0.987±0.001	0.988±0.002	0.990±0.001	0.987±0.001	0.990±0.001	0.992±0.001

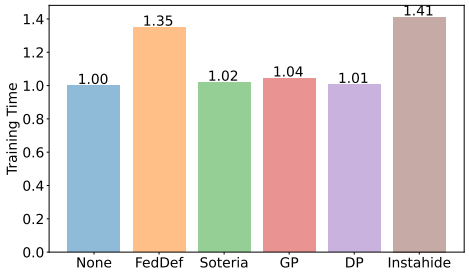


Fig. 9: Relative training time comparison among baselines.

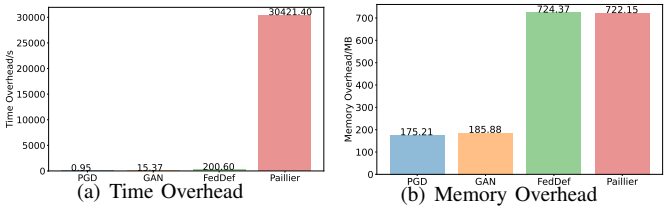


Fig. 10: Absolute time and memory overhead comparison between FL (FedDef and Paillier) and adversarial attack (PGD and GAN).

However, when def_lr is 0.2, more optimization steps can induce privacy loss (only 0.6 score for $def\_ep = 100$ and 0.07 reconstructed label accuracy) during early stage (see Fig. 8) which means pseudo data is over-optimized.

Due to the above consideration, we recommend predetermined constants $\epsilon = 0$, $\delta = 1$, $g\_value = 1e-15$, and key parameter $\alpha = 1$, $def\_lr = 0.2$ and $def\_ep = 40$ as our optimal parameter combination, where the model accuracy is maintained and privacy score is high enough to mitigate any reconstruction attack and the following adversarial attack.

E. Defense Overhead

Finally, we conduct experiments to present defenses' computation and memory overhead.

Setup 7. We first evaluate algorithm running time for all the defenses, where only one local user trains a global DNN model on KDD99 dataset with full training dataset for 100 epochs. To further demonstrate FedDef's advantage, we also compare FedDef alone with traditional homomorphic encryption's (HE) performance in terms of computation and memory overhead.

Specifically, we adapt from an open-source github project [44] named phe, which implements the Paillier Partially Homomorphic Encryption. It first generates both public and private key, where local user encrypts their gradients with public key and decrypts the aggregated gradients with private key. Such encryption and decryption can induce extra overhead during FL. We also follow the same training setting as other defenses and record the absolute training time and memory.

For complete results, we also present the privacy evaluation overhead that includes GAN training (100 epochs) for black-box attack and PGD optimization (100 steps) for white-box attack. For all the experiments, we repeat the process for five times and report the average results.

Overhead Analysis. Fig. 9 illustrates the relative training time under our optimal parameters. We can find that our defense induces additional 0.35 time in exchange for model performance and privacy protection guarantee. Such computation overhead is reasonable because optimization-based defense iteratively searches for optimal pseudo data while other defenses such as differential privacy directly injects noises and does not induce additional computation.

We also present Paillier's overhead along with our privacy evaluation process in Fig. 10. In general, FL process (FedDef and Paillier) can take up more time and memory than adversarial attack (PGD and GAN). Specifically, Paillier is significantly worse and induces about 150 times extra training time ($3,0421s$) than FedDef ($200s$), while our privacy evaluation is also feasible to perform and only takes $< 1s$ for white-box PGD attack and around $15s$ even for GAN-based attack. As for memory occupation, FedDef and Paillier share similar results (722-724 MB), while PGD and GAN-based attack only take up 175-185 MB memory.

The results above demonstrate that our defense has great advantage over HE-based methods in terms of time overhead, and our privacy evaluation (white-box or GAN-based) is feasible to perform that only requires limited resources.

VII. Discussion

1) Additional computation overhead: The main reason for additional training time is that our defense transforms the data for every local update with high gradient computation overhead in Eq. (9). Our future work can try improving Algorithm 1 by transforming private data only once or distilling the model parameters to reduce pseudo gradient optimization.

2) Traffic space attack: For black-box attack, we have only leveraged GAN to generate adversarial features instead of real traffic data to attack the NIDSs. However, generating more practical traffic is beyond the scope of our work, and we can adapt current approaches like GAN+PSO [33] to further improve the attacks and thus better evaluate the defenses.

VIII. Conclusions

In this work, we first propose two privacy metrics specifically designed for FL-based NIDS, i.e., privacy score from reconstruction attacks and evasion rate from GAN-based adversarial attack, to derive an accurate evaluation of privacy protection and demonstrate the insufficiency of existing defenses.

To build a more robust FL-based NIDS, we further propose a novel input perturbation optimization defense strategy, i.e., FedDef, which aims to generate pseudo data to maintain model utility and constrain gradient deviation to provide privacy protection. We also give theoretical analysis for utility and privacy guarantee with our defense. The experimental results illustrate that our defense outperforms existing defenses and proves great privacy protection during both training stages against both attacks while also maintaining model accuracy loss within 3%.

REFERENCES

[1] Y. Zhao, K. Xu, J. Chen, and Q. Tan, "Collaboration-Enabled Intelligent Internet Architecture: Opportunities and Challenges," *IEEE Network*, vol. 36, no. 5, pp. 98–105, 2022.

[2] D. Wagner, D. Kopp, M. Wichtlhuber, C. Dietzel, O. Hohlfeld, G. Smaragdakis, and A. Feldmann, "United we stand: Collaborative detection and mitigation of amplification ddos attacks at scale," in *Proceedings of ACM CCS*, 2021, pp. 970–987.

[3] 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," in *Proceedings of IEEE ICDCS*, 2019, pp. 756–767.

[4] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE TIFS*, vol. 17, no. 8, pp. 5615–5624, 2020.

[5] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proceedings of IEEE SP*, 2019, pp. 691–706.

[6] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of ACM CCS*, 2015, pp. 1322–1333.

[7] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" in *Proceedings of NeurIPS*, vol. 33, 2020, pp. 16 937–16 947.

[8] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proceedings of IEEE INFOCOM*, 2019, pp. 2512–2520.

[9] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[10] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of ACM CCS*, 2016, pp. 308–318.

[11] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *Proceedings of IEEE/CVF CVPR*, 2021, pp. 9311–9319.

[12] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv:1802.09089*, 2018.

[13] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv:1610.05492*, 2016.

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

[15] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for iot security attacks," *IEEE IoTJ*, vol. 9, no. 4, pp. 2545–2554, 2021.

[16] T. Dong, S. Li, H. Qiu, and J. Lu, "An interpretable federated learning-based network intrusion detection framework," *arXiv:2201.03134*, 2022.

[17] X. Wang, S. Garg, H. Lin, J. Hu, G. Kaddoum, M. J. Piran, and M. S. Hossain, "Towards accurate anomaly detection in industrial internet-of-things using hierarchical federated learning," *IEEE IoTJ*, 2021.

[18] J. Böhler and F. Kerschbaum, "Secure multi-party computation of differentially private heavy hitters," in *Proceedings of ACM CCS*, 2021, pp. 2361–2377.

[19] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of Theory of cryptography conference*, 2006, pp. 265–284.

[20] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proceedings of NeurIPS*, 2019.

[21] X. Yang, Y. Feng, W. Fang, J. Shao, X. Tang, S.-T. Xia, and R. Lu, "An accuracy-lossless perturbation method for defending privacy attacks in federated learning," in *Proceedings of WWW*, 2022, pp. 732–742.

[22] Y. Huang, Z. Song, K. Li, and S. Arora, "Instahide: Instance-hiding schemes for private distributed learning," in *Proceedings of ICML*, 2020, pp. 4507–4518.

[23] W. Gao, S. Guo, T. Zhang, H. Qiu, Y. Wen, and Y. Liu, "Privacy-preserving collaborative learning with automatic transformation search," in *Proceedings of IEEE/CVF CVPR*, 2021, pp. 114–123.

[24] J. Wu, M. Hayat, M. Zhou, and M. Harandi, "Defense against privacy leakage in federated learning," *arXiv:2209.05724*, 2022.

[25] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," in *Proceedings of IEEE SP*, 2022, pp. 1897–1914.

[26] S. Mehnaz, S. V. Dibbo, R. De Viti, E. Kabir, B. B. Brandenburg, S. Mangard, N. Li, E. Bertino, M. Backes, E. De Cristofaro, *et al.*, "Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models," in *Proceedings of USENIX Security*, 2022, pp. 4579–4596.

[27] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv:2001.02610*, 2020.

[28] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," *arXiv:2112.02918*, 2021.

[29] G. Danner and M. Jelasity, "Fully distributed privacy preserving mini-batch gradient descent learning," in *Proceedings of IFIP DAIS*, 2015, pp. 30–44.

[30] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of ACM CCS*, 2017, pp. 1175–1191.

[31] Y. Sharon, D. Berend, Y. Liu, A. Shabtai, and Y. Elovici, "Tantra: Timing-based adversarial network traffic reshaping attack," *IEEE TIFS*, vol. 17, pp. 3225–3237, 2022.

[32] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating dnn-based traffic analysis systems in real-time with blind adversarial perturbations." in *Proceedings of USENIX Security*, 2021, pp. 2705–2722.

[33] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, "Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors," *IEEE JSAC*, vol. 39, no. 8, pp. 2632–2647, 2021.

[34] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv:1706.06083*, 2017.

[35] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of IEEE CVPR*, 2016, pp. 2574–2582.

[36] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proceedings of ICML*, 2020, pp. 2206–2216.

[37] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proceedings of ICLR*, 2020.

[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of IEEE symposium on computational intelligence for security and defense applications*, 2009, pp. 1–6.

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

[40] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Proceedings of MilCIS*, 2015, pp. 1–6.

[41] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2014.

[42] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of IEEE SP*. Ieee, 2017, pp. 39–57.

[43] H. Kim, "Torchattacks: A pytorch repository for adversarial attacks," *arXiv:2010.01950*, 2020.

[44] C. Data61, "Python paillier library," https://github.com/data61/python-paillier, 2013.