FedDLM: A Fine-Grained Assessment Scheme for Risk of Sensitive Information Leakage in Federated Learning-based Android Malware Classifier

Changnan Jiang
Beijing Key Lab. of Network Technology
Beihang University
Beijing, China
jcnby@buaa.edu.cn

Chunhe Xia
Beijing Key Lab. of Network Technology
Beihang University
Beijing, China
Guangxi Collaborative Innovation Center of Multi-source Information Integration and Intelligent Processing
Guangxi Normal University
Guilin, China
xch@buaa.edu.cn

Chen Chen
Beijing Key Lab. of Network Technology
Beihang University
Beijing, China
chenchenl@buaa.edu.cn

Huacheng Li
Beijing Key Lab. of Network Technology
Beihang University
Beijing, China
leehc@buaa.edu.cn

Tianbo Wang*
School of Cyber Science and Technology
Beihang University
Beijing, China
Shanghai Key Laboratory of Computer Software Evaluating and Testing,
Shanghai, China
wangtb@buaa.edu.cn

Xiaojian Li
College of Computer Science and Information Technology
Guangxi Normal University
Guilin, China
xiaojian@mailbox.gxnu.edu.cn

***Abstract*—In the traditional centralized Android malware classification framework, privacy concerns arise as it requires collecting users' app samples containing sensitive information directly. To address this problem, new classification frameworks based on Federated Learning (FL) have emerged for privacy preservation. However, research shows that these frameworks still face risks of indirect information leakage due to adversary inference. Unfortunately, existing research lacks an effective assessment of the extent and location of this leakage risk. To bridge the gap, we propose the FedDLM, which provides a fine-grained assessment of the risk of sensitive information leakage in an FL-based Android malware classifier. FedDLM estimates attackers' theoretical maximum inference ability from the information theory perspective to gauge the degree of leakage risk in the classifier effectively. It precisely identifies critical positions in the shared gradient where the leakage risk exists by utilizing characteristics of class activation in classifiers. Through extensive experiments on the Androzoo dataset, FedDLM demonstrates its superior effectiveness and precision compared to baseline methods in evaluating the risk of sensitive information leakage. The evaluation results provide valuable insights into information leakage problems in classifiers and targeted privacy protection methods.**

Keywords—Malware Classification, Leakage Risk, Federated Learning

I. Introduction

Due to the wide use of the Android system and its open source, Android users are frequently infected and attacked by various malware. To deal with security threats, scientists use various **M**achine **L**earning （ML） methods to train malware classifiers, but the training process relies on numerous user app samples through the centralized collection. However, the training app samples uploaded by users contain much sensitive information about users, such as user preferences and device security status [1]. If a curious server directly analyzes the user's app samples, it will analyze the user's sensitive information and violate their privacy [2]. Recent scandals over the misuse of user data by operator and the introduction of numerous data privacy laws have limited the use of malware classifiers based on centralized ML[3]. In addition, with the popularity and evolution of sizeable intelligent analysis systems such as ChatGPT, users have raised privacy concerns about a central server with powerful model reasoning capabilities. The underlying purpose of the central server is difficult to know and control. To solve users' privacy concerns, security experts have begun to focus on the privacy-preserving malware classification framework based on **F**ederated **L**earning (FL) [4]. In the FL-based classifier, users only share the model parameters of the classifier for co-training and do not need to upload their private app samples to the server, avoiding the direct leakage of sensitive information [3].

However, some methods of privacy inference attacks against the FL framework have emerged recently, exposing it to the risk of sensitive information leakage. Typical inference attack methods include model inversion and attribute inference attacks [5]. These attacks have proven to be a significant privacy threat

* Corresponding author: Tianbo Wang

to classification frameworks for image recognition and natural language processing tasks. However, the existing FL-based malware classifier does not consider and evaluate the impact of inference attacks. It does not propose defense schemes, which exposes it to the threat of sensitive information leakage and violates existing privacy protection laws. For AI projects, the GDPR requires a **D**ata **P**rotection **I**mpact **A**ssessment (DPIA) [6]: identifying potential privacy threats in the classification framework and assessing how much it affects user privacy.

However, existing assessment schemes cannot fine-grained meter the degree of sensitive information leakage risk in FL-based Android malware classifiers; and cannot capture the essential reason and location of the leakage risk. In addition, the corresponding sensitive information in the Android malware classifier differs from the reconstructed sensitive features in image or speech tasks (such as pixel or waveform similarity), requiring specific analysis and corresponding metrics design. These shortcomings of the existing methods motivate us to explore an effective and fine-grained leakage risk assessment mechanism for the FL-based Android malware classifier. Therefore, this paper focuses on the following two questions:

- How to effectively meter the risk of sensitive information leakage in the FL-based Android malware classifier?
- How to precisely locate the critical position where the risk of sensitive information leakage exists in the FL-based Android malware classifier?

Our response to these questions is FedDLM, a fine-grained assessment scheme for sensitive information leakage risk in a FL-based Android malware classifier. Specifically, first, we attribute the risk of leakage to the **M**utual **I**nformation (MI) about sensitive information in the shared gradient; Furthermore, we designed a meter mechanism based on MI, LeakMeter, to quantify the degree of sensitive information leakage risk. Then, according to the characteristics of class activation of neural networks and the essential rule of attacker reasoning, we design a risk localization mechanism based on class activation, LeakLocation, to precisely locate the critical position with leakage risk in the shared gradient. The main contributions of this paper are as follows:

- According to the threat characteristics and user privacy requirements of the FL-based Android malware classifier, an assessment scheme FedDLM for leakage risk of sensitive information is proposed. The framework can fine-grained evaluate the degree and critical position of Android users' sensitive information leakage risk. It is the first work to fine-grained assess the leakage risk of users' sensitive information in a FL-based Android malware classifier.
- We propose LeakMeter, which captures the attacker's maximum theory-reachable reasoning ability through the mutual information of sensitive information and gradient in the classifier to effectively meter the risk of sensitive information leakage. We propose LeakLocation, which leverages the properties of class activation in neural network and the reasoning principle of the attacker to precisely locate the critical locations of sensitive information leakage risk in the classifier.
- Extensive numerical simulations of various baseline FL-based Android malware classifiers and inference attack models are carried out on the classic Androzoo dataset. Compared with the baseline assessment schemes, the experimental results show that the proposed FedDLM can more effectively meter the leakage risk of classifiers and more precisely locate the critical leakage risk's position. In addition, the experimental results provide more in-depth theoretical support for developers to study the fine-grained mitigation measures of sensitive information leakage in classifiers.

The rest of this paper is organized as follows. In Section II, we outline the related works. Section III briefly introduces the study's system model and problem description. We present the proposed scheme in Section IV. Experimental evaluation is conducted in Section V. Finally, we discuss our work in Section VI and summarize the work in Section VII.

II. Related Work

A. *FL-based Android Malware Classifier*

The FL-based malware classifier has been widely studied to address users' privacy concerns about traditional centralized malware classifiers. Narendra [7] used the image features of malware and auxiliary classifier generative adversarial network (AC-GAN) to design a FL-based lightweight malware detection model based on CNN. To resist the challenges of data manipulation by adversaries and FL communication energy consumption, RAPID, proposed by Shukla [8], can defend against the security risks of data manipulation by adversaries by identifying relevant local features corresponding to attack strategies. Hsu [3] proposed a FL-based malware detector, which uses permissions and API calls as features and applies support vector machine (SVM) to classify Android apps. Taheri [9] proposed a FL-based malware classifier Fed-IIoT against poisoning attacks, which improved the robustness against poisoning attackers in the system through a GAN network-based defense algorithm (A3GAN). Valerian [10] proposed a robust malware detection mechanism using the FL aggregation method of mean pruning to cope with the threat of label flipping and model cancellation attacks.

However, the existing FL-based malware classifiers only focus on the robustness against poisoning attacks and ignore the consideration and research of sensitive information leakage. It prevents the current scheme from ensuring user data privacy and hinders users' willingness to participate in federal training of large models.

B. *Inferential Attack Methods on FL-based Framework*

Although the FL training method avoids the risk of exposing sensitive information, the adversary can still use various privacy inference attack methods to reason about the shared parameters, resulting in indirect leakage of sensitive information. Four types of existing inference attacks threaten the FL-based learning framework: model inversion attack, model extraction attack, membership inference attack, and attribute inference attack. The attribute inference attack was first proposed by Ateniese [11], which collects shared gradients with different sensitive attributes to train the corresponding meta-classifier to infer the existence of sensitive attributes contained in the user's training dataset. Tramer [12] first proposed a model stealing attack against classification models of black boxes. Tramer constructs a training set by continuously querying the output features of the target model and then trains a similar model. The article [13] first proposed model inversion for classification tasks in the medical domain. Then, a model inversion mechanism is proposed in [14], which can reconstruct key samples for each type of model. Shokri [15] proposed the first membership inference attack method against ML models. They simulate the behavior of the target model by training multiple shadow models and then infer whether the target sample belongs to the training data set of the target model. These inference attacks have been shown to pose significant privacy threats in the image recognition and natural language processing domains.

However, considering the scenario characteristics of the Android malware classifier and users' privacy requirements, there is a lack of evaluation for the actual impact of inference attacks on it.

C. *Leakage Risk Assessment for Federal Learning*

In the article [16], the author used the mean square error between the original graphics and reconstructed graphics to meter information leakage. However, its assessment results depend on the model and ability of the attack, and it is only suitable for scenarios sensitive to reconstructed input features, such as graphic or speech recognition tasks. The article [17] quantifies the leakage risk by estimating the posterior probability of a sample in the target model's training dataset. But the posterior measure is not positively correlated with the adversary's ability to infer sensitive information. In the article [18], the authors used IRFIL to estimate the privacy risk of members. However, this method is only suitable for capturing the relevant information between the sample and the model and cannot calculate the content of sensitive information in the sample. In Article [19], the author proposed the maximum information leakage as a metric of the leakage risk of members, which quantifies the leakage of all information in the data but cannot distinguish the leakage of sensitive information and non-sensitive information. Paper [20] proposed instance's privacy accounting method for the DP-SGD training model. However, privacy accounting method is based on calculating privacy on a single instance, which is only suitable for membership inference but not for evaluating attribute or label inference.

In summary, the existing evaluation methods only rely on empirical attack effects or are based on side information (such as model utility or sensitivity) and do not capture the essential reasons for sensitive information leakage. Therefore, designing an effective and fine-grained assessment scheme is necessary for the risk of sensitive information leakage in the FL-based Android malware classifier.

III. System Model and Problem Description

A. *System Model of a FL-based Android Malware Classifier*

In a typical FL-based Android malware classifier system, $\mathcal{C}$ Android clients collaborate to train the classification model in the organization of an FL server. Training app samples of client are stored locally and not exchanged [3]. The FL server coordinates the collaborative training process by iteratively repeating steps until convergence of the FL-based Android malware classification model. In the initialization stage, FL server assigns an initialized global classification model w^t to each Android client. Then, each client trains and updates the current w^t by using the local dataset $\mathcal{D}_c$ of D_c size in round t. In the aggregation stage, FL server receives local gradients of client. The local loss function $\mathcal{J}_c(w)$ and global loss function $\mathcal{J}(w)$ to be optimized in system are shown in (1)-(2):

$$\mathcal{J}_c(w) = \frac{1}{\mathcal{S}_c} \sum_{z_i \in D_c} f_c(w; z_i) \tag{1}$$

$$\min_{w \in R^d} \mathcal{J}(w) = \sum_{c=1}^{c=\mathcal{C}} \left[\frac{\mathcal{S}_c}{\mathcal{S}} \cdot \mathcal{J}_c(w)\right] \tag{2}$$

Where $z_i = (x_i, \mathcal{L}_i, \mathcal{T}_i), \forall i \in [1, \dots, \varphi_c]$ is sampled from the D_c of c clients, $c \in [1, \mathcal{C}]$; $\mathcal{S}$ is the total quantity of global samples, $\mathcal{S}_c$ is the quantity of samples of client c ; $x_i \in \mathbb{X}$ is the feature of Android malware, such as APIs and Permissions of Android app; $\mathcal{L}_i \in \mathbb{L}$ is the category label of Android malware, such as Exploit and Spyware; $\mathcal{T}_i \in \mathbb{T}$ is the category of application functions of app sample z_i, such as Finance and Weather; $f_c(\cdot)$ is the loss function of Android client c.

$$w^{t+1} \leftarrow w^t + \sum_{c=1}^{c=\mathcal{C}} \left(\frac{\mathcal{S}_c}{\mathcal{S}} \cdot \nabla_{w^t} \mathcal{J}_c^t\right) \tag{3}$$

In the update stage, FL server uses the federated average algorithm (Fedavg) [3] to generate a new global model w^{t+1} for the next round, as shown in (3). Where $\mathcal{G}_{t,c} = \nabla_{w^t} \mathcal{J}_c^t$ denotes the shared gradient of c client.

B. *Threat Model of a FL-based Android Malware Classifier*

The adversary considered in this paper is a curious and honest FL server. It honestly follows the training protocol of the FL-based Android malware classifier to organize users for joint training but is curious about sensitive information in the client user data. In addition, the communication between the server and the client is confidential to the outside, and poisoning and active attacks are not considered. To provide users with the worst-case leakage risk to ensure sufficient privacy protection (based on the point of view of information-theoretic privacy), we set the inference ability of the adversary:

- The adversary (the FL server) obtains all parameters and gradients of the user's model.
- The adversary has enough auxiliary datasets with the same distribution as the client to train the inference model, and the computational power is not constrained.
- The adversary can use a variety of inference attacks (four types) to infer the sensitive information that violates the user's privacy in the training data.

C. *Design Goal of an Assessment Scheme*

This paper aims to provide users with a practical assessment scheme for sensitive information leakage risk so that users can understand the degree of leakage risk and the critical position of the leakage risk. Specifically, the designed assessment scheme should meet the following two goals:

Goal 1: The degree of risk metered should be independent of the attack model and track the attacker's theoretical maximum achievable inference ability. The attack models' type are endless, and the user's perspective cannot know the attacker's methods and capabilities. Therefore, the assessment scheme should capture the essential reasons for sensitive information leakage and meter the risk based on the attacker's maximum inference ability to provide users with objective and effective results.

Goal 2: Critical locations of leakage risk should be assessed precisely. In the FL-based Android malware classifier (neural network), the shared gradient parameters are high-dimensional, and the mapping relationship with sensitive information is complex and nonlinear. It makes users difficult to precision understand the source of the leak and take practical defensive actions. Therefore, the assessment scheme needs to be precision located where the leakage risk exists to obtain useful results.

IV. Proposed Methods

In this section, we first give an overview of FedDLM, and then introduce the functionality and design details of its sub-components, LeakMeter and LeakLocation, respectively.

A. Overview of Proposed FedDLM

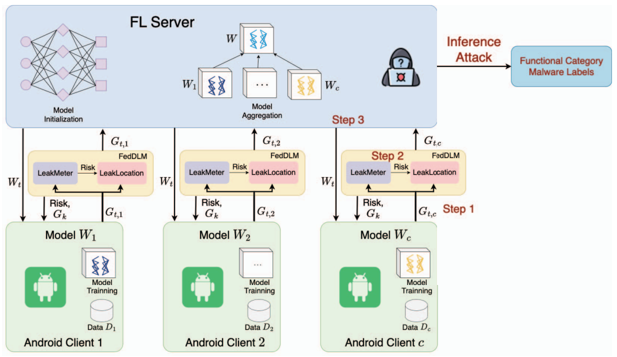


Fig. 1. Overview of proposed FedDLM.

The overall framework of FedDLM is shown in Fig. 1. It includes LeakMeter, a metering mechanism for the risk of sensitive information leakage, and LeakLocation, a locating mechanism for the risk of sensitive information leakage. The workflow of the FedDLM includes the following three steps:

Step 1: The client accepts the global model w^t from the FL server for local training. Then, the gradient $\mathcal{G}_{t,c}$ of this round is sent to FedDLM for meter risk of sensitive information leakage.

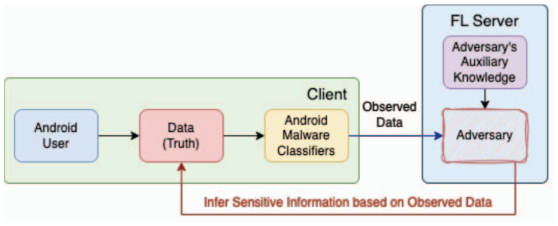
Step 2: LeakLocation uses the risk score calculated in LeakMeter to locate the key location G_k in gradient $\mathcal{G}_{t,c}$ where the risk of sensitive information leakage exists. Then, $Risk^t$ and G_k is sent to the client, and $\mathcal{G}_{t,c}$ is uploaded to the FL server.

Step 3: The FL server receives the $\mathcal{G}_{t,c}$, summarizes it, and updates the global model w^{t+1}. Then perform the first step and loop until the model converges.

B. Design of LeakMeter

1) Design of Leakage Risk Score in LeakMeter

To effectively meter the degree of risk of sensitive information leakage, the elements used to calculate the risk score should be relevant variables that reflect the essential cause of the leakage in the system. The principle of sensitive information leakage is shown in Fig. 2: In a given Android malware classifier, an attacker can infer users' sensitive information by analyzing the observable data sources (generated from training data that contains sensitive information) users share, using auxiliary knowledge and inference attack models. Therefore, sensitive information and corresponding observable data sources are used to calculate the risk score. Considering users' privacy requirements and the characteristics of the FL-based Android malware classifier, we define and analyze the targeted sensitive information and observable data sources in detail.



FL Server
Adversary's Auxiliary Knowledge
Client
Observed Data
Android User
Data (Truth)
Android Malware Classifiers
Adversary
Infer Sensitive Information based on Observed Data

Fig. 2. The principle of sensitive information leakage.

Definition 1 (Sensitive information). *Sensitive information is the information that an attacker may infer in the training sample of the FL-based Android malware classifier to violate the user's attribute privacy.*

According to the definition of user privacy, it is divided into identity privacy and attribute privacy. The client's ID number representing the user's identity is known in the FL classifier. Therefore, in this paper, we focus on the attribute privacy of the user, that is, the ability of an individual whose identity is known to hide its attribute information, such as the user's usage preferences and security status. In the federal Android malware classifier and its corresponding threat model, the input features (APIs or Permissions of Android app) are not sensitive, and its reconstruction or membership inference cannot reflect attribute information about the user; The parameters of the classification model are common to all clients, and the information of specific clients cannot be inferred. In addition, the function category $\mathcal{T}$ of the Android app obtained by the attribute inference attack reveals the installation/use information of the user; The tags $\mathcal{T}$ of Android malware obtained by tag inference attacks reflects the security status information of the user's device, both of which violate the user's attribute privacy. After a comprehensive analysis, the sensitive information that the attacker can actually infer in the FL-based Android malware classifier is $\mathcal{V} = \{\mathcal{L}, \mathcal{T}\}$.

The data processing and prediction stages are carried out on the client in a closed manner, and the attacker cannot obtain the observable data related to sensitive information. However, during the training phase, the FL server can obtain the client's shared gradient $\mathcal{G}_{t,c}$ (trained by sensitive information data). In addition, we consider a sampling mechanism with no putback, where the opponent can only infer the gradient for each training turn individually. Thus, during the life cycle of the FL-based Android malware classifier, the adversary's observable data source is the shared gradient $\mathcal{G}_{t,c}$ of each round of training.

According to the principle of information leakage, the inference attacks' essential principle is to observe the gradient change rule to infer the existence of sensitive information. From an information-theoretic point of view, the essential reason for inferring success is because the gradient contains the "information" of sensitive information. The greater the MI between the shared gradient and the sensitive information, the more significant the dependence between them, and the easier it is for an attacker to infer the sensitive information in the gradient. Therefore, we use the **N**ormalized mutual **I**nformation (NI) between the sensitive information $\mathcal{V}$ (total $s+t$ categories) and the shared gradient $\mathcal{G}_{t,c}$ as the risk score, calculated as (4):

$$Risk_{\mathbb{S},\mathcal{V}}^{t,c} = \frac{\sum_{r=1}^{s+t} NI(\mathcal{V}_r;\mathcal{G}_{t,c})}{s+t} \tag{4}$$

Where $Risk_{\mathbb{S},\mathcal{V}}^{t,k}$ represents the degree of leakage risk of the FL-based Android malware classifier $\mathbb{S}$ for the $\mathcal{V}$ in the training dataset at round t. $NI(\mathcal{V}_r;\mathcal{G}_{t,c})$ represents the NI between the $\mathcal{V}$ and shared gradients $\mathcal{G}_{t,c}$. The average value of $NI(\mathcal{V}_r;\mathcal{G}_{t,c})$ is taken to indicate the overall degree of leakage risk in classifier $\mathbb{S}$. If $Risk_{\mathbb{S}_A,\mathcal{V}}^{t,c} > Risk_{\mathbb{S}_B,\mathcal{V}}^{t,c}$, then the leakage risk of the classifier $\mathbb{S}_A$ is greater than that of $\mathbb{S}_B$ to the corresponding sensitive information $\mathcal{V}$. $Risk_{\mathbb{S},\mathcal{V}}^{t,c}$ objectively captures the maximum reasoning power an attacker can achieve to meter the risk of leakage in the system, so it satisfies **Goal 1**. The overall flow of the LeakMeter is shown in **Algorithm1.**

Algorithm1: Leakage risk meter mechanism: LeakMeter.

```
Input:   Gradient of client $c$ in round $t$ is $\mathcal{G}_{t,c}$, the sensitive
         information $\mathcal{V}$, sample $\{(\mathcal{V}_i,\mathcal{G}_i)\}_{i=1}^N$ from $p(\mathcal{V},\mathcal{G})$.
Output:  Leakage risk score $Risk_{\mathbb{S},\mathcal{V}}^{t,c}$ of the gradient $G_{t,c}$.
1:  For $r = 1,\cdots,s+t$ do
2:      Calculate $\ell(\theta) = \frac{1}{N_r}\sum_{i=1}^{N_r} \log q_{\mathcal{H}_\theta}(\mathcal{V}_{r,i} \mid \mathcal{G}_i)$;
3:      Training $q_{\mathcal{H}_\theta}(\mathcal{V} \mid \mathcal{G})$ by maximizing $\ell(\theta)$;
4:      Calculate $\widehat{\mathrm{MI}_{\mathrm{vCLUB}}}(\mathcal{V}_r;\mathcal{G}) = \frac{1}{N_r}\sum_{i=1}^{N_r}[\log q_{\mathcal{H}_\theta}(\mathcal{V}_{r,i} \mid \mathcal{G}_i)$
        $-\frac{1}{N_r}\sum_{j=1}^{N_r} \log q_{\mathcal{H}_\theta}(\mathcal{V}_{r,i} \mid \mathcal{G}_i)], i \neq j$;
5:      Calculate normalized $NI(\mathcal{V}_r;\mathcal{G}) = \frac{\widehat{\mathrm{MI}_{\mathrm{vCLUB}}}(\mathcal{V}_r;\mathcal{G})}{min\{H(\mathcal{V}_r),H(\mathcal{G})\}}$;
6:  End For
5:      Calculate $Risk_{\mathbb{S},\mathcal{V}}^{t,c} = \frac{\sum_{r=1}^{s+t} NI(\mathcal{V}_r;\mathcal{G}_{t,c})}{s+t}$;
6:  Return $Risk_{\mathbb{S},\mathcal{V}}^{t,c}$;
```

2) *Design of Calculation Method of Leakage Risk Score*

The expression of MI between $\mathcal{V}$ and the $\mathcal{G}$ is shown in the (5). However, the mapping relationship between $\mathcal{G}$ and $\mathcal{V}$ is high-dimensional and nonlinear, and $p(\mathcal{V} \mid \mathcal{G})$ are unknowable. Therefore, the traditional MI estimation methods, such as kernel density and likelihood ratio estimation, which are suitable for estimating low-dimensional and linear mapping relations, cannot provide reliable estimation for the system.

$$MI\ (\mathcal{V};\mathcal{G}) = \mathbb{E}_{p(\mathcal{V},\mathcal{G})}[log\ \frac{p(\mathcal{V}|\mathcal{G})}{p(\mathcal{V})}] \tag{5}$$

Therefore, to estimate the complex and high-dimensional $MI\ (\mathcal{V};\mathcal{G})$, we use the MI estimation in the form of **v**ariational **C**ontrastive **L**og-ratio **U**pper **B**ound (vCLUB) in the literature [21]. Furthermore, since the conditional distribution $p(\mathcal{V} \mid \mathcal{G})$is unknown in the gradient, we approximate the true $p(\mathcal{V} \mid \mathcal{G})$ by using the variational distribution $q_{\mathcal{H}_\theta}(\mathcal{V} \mid \mathcal{G})$ expressed by the neural network ($\mathcal{H}_\theta: \mathcal{G} \times \mathcal{H}_\theta \to \mathcal{V}$). The upper bound estimate $MI_{\mathrm{vCLUB}}\ (\mathcal{V};\mathcal{G})$ of $MI\ (\mathcal{V};\mathcal{G})$ is shown in the (6):

$$\begin{aligned} MI_{\mathrm{vCLUB}}\ (\mathcal{V};\mathcal{G}) = \mathbb{E}_{p(\mathcal{G},\mathcal{V})}\left[\log q_{\mathcal{H}_\theta}(\mathcal{V} \mid \mathcal{G})\right] - \\ \mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p(\mathcal{V})}\left[\log q_{\mathcal{H}_\theta}(\mathcal{V} \mid \mathcal{G})\right] \end{aligned} \tag{6}$$

When $q_{\mathcal{H}_\theta}(\mathcal{V} \mid \mathcal{G})$ is sufficient to approximate $p(\mathcal{V} \mid \mathcal{G})$, $MI_{vCLUB}(\mathcal{V};\mathcal{G})$ obtain upper bound of $MI\ (\mathcal{V};\mathcal{G})$. In addition, vCLUB connects MI estimation with contrast learning and estimates MI using the $p(\mathcal{V} \mid \mathcal{G})$ difference between positive and negative sample pairs. This linear form improves the numerical stability of calculations for high-dimensional MI estimates and makes it easier to calculate. Using the samples $\{(\mathcal{V}_i,\mathcal{G}_i)\}_{i=1}^N$, the unbiased estimator $\widehat{MI_{vCLUB}}(\mathcal{V};\mathcal{G})$ of $MI_{vCLUB}(\mathcal{V};\mathcal{G})$ is shown as (7):

$$\begin{aligned} \widehat{MI_{vCLUB}}(\mathcal{V};\mathcal{G}) = \frac{1}{N}\sum_{i=1}^{N}[\log\ q_{\mathcal{H}_\theta}(\mathcal{V}_i \mid \mathcal{G}_i) - \\ \frac{1}{N}\sum_{j=1}^{N} \log q_{\mathcal{H}_\theta}(\mathcal{V}_j \mid \mathcal{G}_i)] \end{aligned} \tag{7}$$

Where, $\{\log q_{\mathcal{H}_\theta}(\mathcal{V}_j \mid \mathcal{G}_i)\}_{i\neq j}$ provide the conditional log-likelihood of negative sample pair $(\mathcal{V}_j,\mathcal{G}_i)$. The range of the $MI(\mathcal{V};\mathcal{G})$ is $\{0, min[H(\mathcal{V}),H(\mathcal{G})]\}$. Further, we define the normalized NI as shown in the (8):

$$NI(\mathcal{V};\mathcal{G}) = \frac{MI(\mathcal{V};\mathcal{G})}{min\{H(\mathcal{V}),H(\mathcal{G})\}} \tag{8}$$

Where, $H(\mathcal{V}) = -\sum_{r=1}^{s+t} p(\mathcal{V}_r)\ log_2(p(\mathcal{V}_r)) \leq log_2(s+t)$ (Jensen inequality), $H(\mathcal{G}) = -\int_{\mathcal{G}}\{f(\mathcal{G})\ log_2(f(\mathcal{G}))\}\,d\mathcal{G}$.

C. Design of LeakLocation

Because of the large scale of neural network parameters, there are complex nonlinear relations between neurons, and the training process is random. It makes it difficult to understand the specific meaning of different elements in the model's parameters or gradients to determine where sensitive information exists in the high-dimensional gradient (used by the attacker to infer). Therefore, we study how to locate the sensitive information in the gradient of a specific model from the perspective of the interpretation method of the neural network model.

Existing interpretive methods propose the widespread existence of objective phenomena of class activation or class selectivity [22] in neural networks, providing valuable insights for localizing sensitive information. The property of class activation is generally understood as the preference of the neural network unit for the class of the input instance. Specifically, certain weights or neurons of the network are activated more strongly when they encounter examples of a given class. Similarly, inputting certain types of sensitive information can make the gradient fluctuate more at specific locations. At the same time, the inference attack model extracts relevant information by observing the change rule of specific locations caused by the appearance of sensitive information. Therefore, combined with the above analysis, we locate the critical source of sensitive information leakage risk in the part of the gradient with a substantial degree of class activation of sensitive attributes. We define the intensity of class activation of the unit in the gradient as Class Influence to search for crucial locations where risks exist.

Definition 2 (Class Influence). *Given the function used to calculate the gradient $G : D \to R^d$, and $G^{m,n}$ is the unit of G; for any two adjacent datasets with different presence of sensitive information $D_{\mathcal{V}}$ and $D_{\bar{\mathcal{V}}}$,define the **Class Influence** $\Delta_2 G_{\mathcal{V}}^{m,n}$ as (9):*

$$\Delta_2 G_{\mathcal{V}}^{m,n} = \max_{D_{\mathcal{V}}, D_{\bar{\mathcal{V}}}} \parallel G^{m,n}(D_{\mathcal{V}}) - G^{m,n}(D_{\bar{\mathcal{V}}}) \parallel_2 \qquad (9)$$

Where, $\parallel G^{m,n}(D_{\mathcal{V}}) - G^{m,n}(D_{\bar{\mathcal{V}}}) \parallel_2$ is the $L2$ distance between $G^{m,n}(D_{\mathcal{V}})$ and $G^{m,n}(D_{\bar{\mathcal{V}}})$. Class Influence measures the extent to which the input of a particular class of sensitive information can cause a change in the gradient of the output. To find the critical position where the risk of sensitive information leakage exists in the gradient, the part with large Class Influence, we sort the value of Class Influence $\Delta_2 G_{\mathcal{V}}^{m,n}$ of each unit $G^{m,n}$ in the G. Then we select the local gradient G_k composed of the κ elements with the highest Class Influence as the critical position of the risk source, as shown in the (10):

$$G_k = \underset{G^{m,n} \subseteq G}{Top\kappa} \left\{\Delta_2 G_{\mathcal{V}}^{m,n}\right\} \qquad (10)$$

In addition, the gradients are sparse [23] in the malware classification model. It means that the gradient fraction with strong class activation is tiny. Therefore, we give the precision metric $\mathcal{P}$ of the location of the risk to search the critical position where the leakage risk exists effectively, as shown in (11):

$$\mathcal{P} = \frac{\mathscr{g}}{\kappa} \cdot \frac{NI(\mathcal{V};\mathcal{G}_k)}{NI(\mathcal{V};\mathcal{G})}, \mathcal{G}_k = G_k(D) \qquad (11)$$

Where $\mathscr{g}$ is the total dimension of the gradient $\mathcal{G}$, and $\frac{NI(\mathcal{V};\mathcal{G}_k)}{NI(\mathcal{V};\mathcal{G})}$ is the ratio of $\mathcal{G}_\kappa$ to the risk of leakage of $\mathcal{V}$ in $\mathcal{G}$. High location precision means that finding the smallest possible area has the most significant ratio of leakage risk, so that users can understand the source of leakage clearly. So it satisfies **Goal 2** and is convenient for efficient and targeted privacy defense. The overall flow of the LeakLocation is shown in **Algorithm2**.

```
Algorithm2: Leakage risk localization mechanism: LeakLocation.
Input:   Calculate function of gradient $G$ , the sensitive informa
         tion $\mathcal{V}$ , data set $D = (D_{\mathcal{V}}, D_{\bar{\mathcal{V}}})$, selected parameters $\kappa$
Output:  Key position of risk in the gradient $G_k$ and precision $\mathcal{P}$
1:       Calculate $\Delta_2 G_{\mathcal{V}}^{m,n} = \max_{D_{\mathcal{V}}, D_{\bar{\mathcal{V}}}} \parallel G^{m,n}(D_{\mathcal{V}}) - G^{m,n}(D_{\bar{\mathcal{V}}}) \parallel_2$
2:       Calculate $G_k = \underset{G^{m,n} \subseteq G}{Top\kappa} \{\Delta_2 G_{\mathcal{V}}^{m,n}\}$;
3:       Calculate $\mathcal{P} = \frac{\mathscr{g}}{\kappa} \cdot \frac{NI(\mathcal{V};\mathcal{G}_k)}{NI(\mathcal{V};\mathcal{G})}, \mathcal{G}_k = G_k(D)$
4:       Return $G_k$ and $\mathcal{P}$;
```

V. Evaluation

The **R**esearch **Q**uestions (**RQs**) below guided the experiments we evaluated.

RQ1: Can the proposed FedDLM effectively meter the leakage risk in a FL-based Android malware classifier? How does it compare to the baseline risk assessment schemes?

RQ2: Can the proposed FedDLM precisely locate the critical locations of leakage risk in a FL-based Android malware classifier? How does it compare to the baseline risk assessment schemes?

A. Experimental Settings

1) Dataset

To verify the FedDLM's validity, we built a dataset from the AndroZoo [24], which has about 59,000 Android apps (nine categories) for the label inference attack and Android malware classification task, as shown in TABLE I. In addition, we extract Android app's functional categories (ten categories) from the metadata in APK files for attribute inference task.

TABLE I. Dataset.

Malware Label	Num	App Functional Category	Num
Benign	29977	Game	4874
Malicious	29932	Books	4861
Adware	3865	Weather	4632
Trojan	3338	Travel and maps	4793
Riskware	4894	Health and Fitness	4842
Ransom	4322	Photography	4665
Exploit	3226	Finance	4745
Spyware	2488	Music and Audio	4441
Downloader	4025	Shopping	4175
Fraudware	3774	Communication	4657

2) The Setup of Attack Model

For the inference test of sensitive information in the federal Android malicious classifier, we choose the label inference attack model proposed in articles [25], [26] and [5] and the attribute inference attack model proposed in articles [27], [5] and [29]. They use share gradients to infer the type of functionality applied in the training samples with the malware label. We follow the model structure set up in the original article. The attacker's auxiliary data set is the entire app data set, using the update gradient generated on the malware classifier from the auxiliary data set to train the attack model and perform testing of the attack (10-fold cross-validation).

3) The Setup of FL-based Malware Classifier

To verify the performance of the proposed risk assessment scheme, we selected two baselines FL-based Android malware classification frameworks, FedIIoT [14] and FedRAPID [16], as the evaluated systems. The classification model for FedIIoT contains seven fully connected layers, using the APIs, Intents, and Permissions as input features. The classification model for FedRAPID contains five convolution layers, followed by four fully connected layers; The gray level image after binary conversion of the application program is used as the input feature. We used the same preprocessing method for the data set as the original article. We built ten clients for FL training with a batch size of 64 and a learning rate of 0.018. We perform the experiment on an Intel Golden 6240 CPU and an NVIDIA A100 GPU. Different risk assessment methods, inference attack models, and FL-based malware classifiers are implemented in

PyTorch. We adopt 10-fold cross-validation to generate test sets from 1/10 user data randomly. The reported results (Android malware classification and inferred attack task) correspond to the average of 20 experiments.

4) Comparing Methods

We compared the performance of FedDLM with the following baseline risk assessment schemes on the tasks of risk degree meter and risk location. IRFIL [18]: Fisher's information for the sample is used as a value for the risk of leakage. Sensitivity [28]: The sensitivity of the gradient to the sample input is used to calculate the degree of leakage risk. At the same time, sensitivity and the size of local gradient blocks are used to divide the risk of different locations. LIP [30]: Calculated the risk level of each weight based on the output value of neurons and weights to locate key risk positions.

5) Evaluation Methods and Metrics

In this part, we introduce the metrics used to evaluate the effect of inference attacks on sensitive information、the effectiveness of leakage risk scoring, and the precision of locating leakage risk. In the actual threat scenario, the ability and means of the attacker are unknown. Therefore, to more comprehensively evaluate the actual threat degree of the adversary's attack (the worst case), we choose the max F1-score of the inference attacks (the average of 20 inference experiments) as the metric of the inference attack effect (the effect of the label and the attribute inference attack are averaged) in the test results of multiple well-turned attack models. The inferred F1-score represents the extent to which inference attacks threaten sensitive information, and we use it as the basis truth value for leakage risk. Therefore, the effective leak risk score should be closely related to the corresponding inferred F1-score (and its changing trend). Therefore, we evaluate the effectiveness of the leakage risk assessment mechanism by calculating the Pearson coefficient between scores of leakage risk in the training process (with different parameter settings of model) and the inferred F1-score. The precision metric for locating the risk of leakage is $\mathcal{P}$, as shown in (11). To better verify the precision of risk location method, we designed additional validation metrics: the ratio of an attacker's inferred F1-score on Top-k elements to all elements $\delta = \frac{F1-score(G_k)}{F1-score(G)}$. The higher the δ value, the higher the precision of risk location.

B. Experimental Results

1) RQ1 – Comparison of The Effectiveness of The Risk Score of Sensitive Information Leakage

To verify the effectiveness of FedDLM's risk score, we systematically assess the leakage risks of the baseline federal malware classifiers FedIIoT [9] and FedRAPID [8] using multiple baseline inference attack models. In the same simulation configuration, compared the effectiveness of the risk score of FedDLM with that of the state-of-the-art IRFIL [18] and Sensitivity [28].

Then, we calculated and plotted the risk scores (in different round) of different evaluation mechanisms and the average F1-score of corresponding inference attacks, as shown in Fig. 3 and Fig. 4. To better display, the risk scores have been properly standardized. Leakage risk score of FedDLM proposed by us is closely related (Pearson coefficient is 0.91) to the F1-score of inference attacks, accurately tracking the changing trend of leakage risk (It rises gradually and then decreases). Sensitive's method tracked the trend of leakage risk to a certain extent, but the correlation was lower than FedDLM, and the Pearson coefficient was only 0.75. It is because Sensitive's method focuses on the sensitivity of the gradient to the training sample and does not calculate the sensitivity of the sensitive information in fine grain. Therefore, it does not capture the essence of the risk of sensitive information leakage, and the score's effectiveness is poor. IRFIL had the worst score (Pearson coefficient of 0.41). Because it only focuses on the sensitivity of the optimal model's weight to the sample but does not fully consider the model’s state (like degree of fit) changes during the training cycle. Therefore, it cannot effectively meter the leakage risk degree of different rounds.

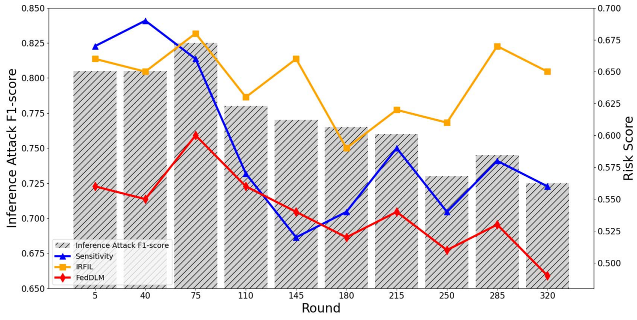


Fig. 3. Scores of leakage risk in different round of training (FedIIot).

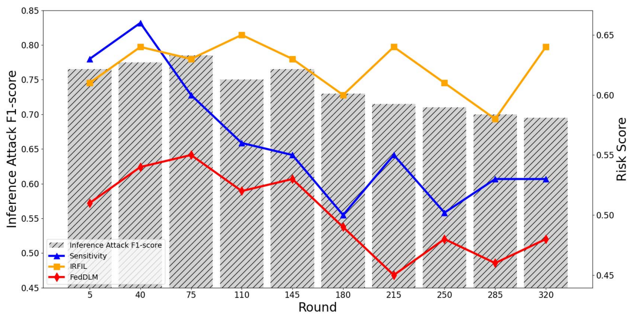


Fig. 4. Scores of leakage risk in different round of training (FedRAPID).

In addition, we set batch size $\beta = [2,4,8,16,32,64,128]$ to construct multiple use cases for testing. As shown in Fig. 5 and Fig. 6, the F1-score of the inference attack continuously decreases as the batch size increases from 2 to 128. It is because as more samples are added to the training, the sensitive information in the computed gradient (average aggregation) of the batch is "diluted" by more non-sensitive information. In addition, more training samples improve the training model's generalization performance and reduce the degree of "memory" of the gradient for sensitive information. Because FedDLM estimated the content of sensitive information from the perspective of information, it accurately captured the process of leakage risk changing with batch size, and the Pearson coefficient was as high as 0.93. On the other hand, Sensitivity and IRFIL methods are calculated based on information from a single sample; Although IRFIL overlays Fisher's information for multiple samples, it does not adequately capture the dilution effect of gradient averaging. Therefore, the effectiveness of the two risk scores in different use cases of batch size change is poor, and the Pearson coefficient is lower than 0.4.

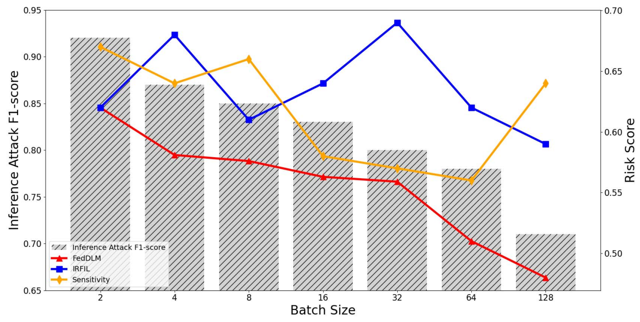


Fig. 5. Scores of leakage risk in different setting of batch size (FedIIot).

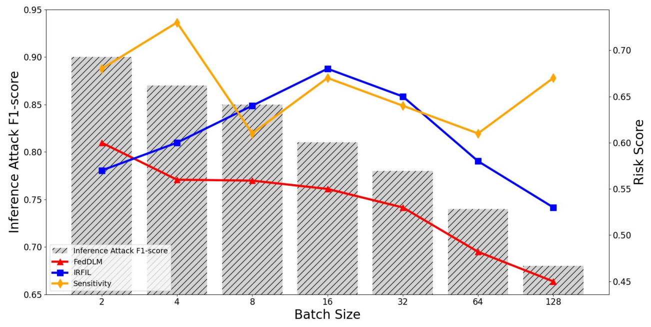


Fig. 6. Scores of leakage risk in different setting of batch size (FedRAPID).

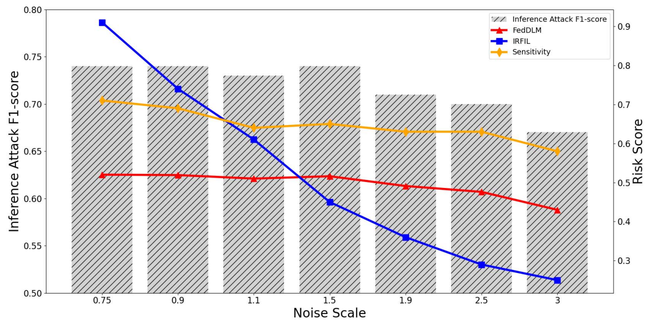


Fig. 7. Scores of leakage risk in different setting of noise scale (FedIIot).

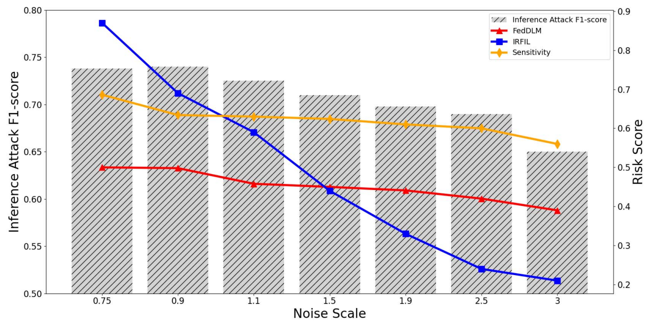


Fig. 8. Scores of leakage risk in different setting of noise scale (FedIIot).

In addition, we set multiple differential privacy (Gaussian) noise scales $\sigma = [0.75,0.9,1.1,1.5,1.9,2.5,3]$ for the gradient to build multiple use cases for testing. As the noise scale increases from 0.75 to 3, the F1-score of the attack decreases continuously, as shown in Fig. 7 and Fig. 8. Because the increase of Gaussian noise interferes with the distribution of sensitive information in the gradient, it prevents the opponent from inferring the sensitive information. FedDLM's score has the highest Pearson coefficient of 0.95 with the inference attacks' F1-score, meaning that FedDLM accurately captures changes in leakage risk. It is because the gradient's sensitivity decreases as the noise increases. However, the Sensitivity's score does not capture the sensitivity of sensitive information leakage, so the measure's effectiveness is slightly worse (a Pearson coefficient of 0.86) than that of FedDLM. In addition, the effectiveness of IRFIL's risk score is poor (Pearson's coefficient is only 0.76), especially in instances with smaller noise scales. It is because the IRFIL's risk score is proportional to $\frac{1}{\sigma}$. However, in the less noisy region $[0.75,1.5]$, the difficulty of the opponent's privacy inference changes little with the noise scale, and the inference ability does not increase infinitely with the noise decrease. As a result, IRFIL's risk score does not effectively assess the risk of leakage for these use cases.

2) *RQ2 – Comparison of the Precision of Risk Location Method*

To verify the effect of FedDLM on risk location, we conducted risk location experiments on different settings of k. In addition, in the same simulation configuration, the effects are compared with the baseline risk location method LIP [30] and Sensitivity [28]. At the same time, the corresponding positioning precision metric $\mathcal{P}$ in different settings and the attack's F1-score ratio δ of the positioned elements are calculated, as shown in the TABLE II. and TABLE III.

FedDLM obtained the best positioning precision $\mathcal{P}$ at different settings of k , and the positioned elements accounted for the highest proportion δ of the attack's F1-score (leakage risk). It shows that FedDLM successfully captures the essential cause of leakage through the difference of elements' Class Influence and precisely identifies the critical elements with the highest risk. The positioning effect of the Sensitivity's method is second. It is because the Sensitivity's method is based on the information of all characteristics in the sample to rank the risk degree and cannot reflect the risk degree of sensitive information. On the other hand, the Sensitivity's method also takes the number of elements as the weight of superposition as the basis of a risk degree. However, the number of elements only increases the reconstruction risk of the complete feature and has a low correlation with the leakage of the implicit sensitive information of the input. Therefore, the positioning effect of the Sensitivity's method is not good. In addition, LIP was the least effective in locating the key risk elements, capturing a lower proportion (lowest $\mathcal{P}$ value, δ only about 25%). It is because the risk location method of LIP is based on the neuron's output size. The output size of the neuron does not reflect the inference rules of the opponent for sensitive information (the opponent considers the change of the neuron output).

To further verify the positioning effect of FedDLM on the risk location, we perform the same configuration inference attack experiment on the remaining gradient, excluding k elements. At the same time, the corresponding positioning precision metric of different k settings and proportion $\gamma = \frac{F1-score(G_{\bar{k}})}{F1-score(G)}$ of inference attack F1-score of the remaining elements are counted, as shown in the TABLE IV. In the two FL-based Android malware classifiers, after removing high-risk elements (more than 240) in gradient, the remaining gradients were attacked with an F1-score below 50% (random guess); and the proportion γ of inferred attack's F1-score of the remaining elements is lower than 50% of the total. It further demonstrates the precision of FedDLM's locating. In addition, we have two findings in the test:

Finding 1: $\gamma + \delta$ is always not equal to 1, but roughly 1.05~1.3, indicating information redundancy between different gradient elements. It is because there is a certain correlation between the gradients of different neural network layers due to the weight connection. However, removing more high-risk elements can keep the risk of leakage at a lower level.

Finding 2: The locating precision $\mathcal{P}$ and δ of different location methods in FedIIoT are higher than that of FedRAPID; At the same time, FedIIoT had a higher F1-score for inferred attacks in different Settings; After removing the high-risk elements, the remaining elements of FedRAPID were attacked at a higher rate than FedIIoT. It shows that in FedRAPID's classification model, the distribution of elements activated by classes with sensitive information is more dispersed and thus less threatened by inference attacks. On the other hand, a more dispersed distribution of elements with class activation makes efficient isolation of high-risk elements more difficult to achieve.

TABLE II. THE PRECISION OF DIFFERENT RISK LOCATION METHODS (FEDIIOT).

Risk Location Mechanism	k =40		k =80		k =160		k =240		k =420	
	$\mathcal{P}$	δ	$\mathcal{P}$	δ	$\mathcal{P}$	δ	$\mathcal{P}$	δ	$\mathcal{P}$	δ
FedDLM	**859**	**0.35**	**568**	**0.47**	**351**	**0.56**	**273**	**0.68**	**169**	**0.76**
LIP	437	0.17	268	0.21	174	0.27	142	0.32	93	0.38
Sensitivity(block)	685	0.26	401	0.32	235	0.36	172	0.39	112	0.45

TABLE III. THE LOCATING PRECISION OF DIFFERENT RISK LOCATION METHODS (FEDRAPID).

Risk Location Mechanism	k =40		k =80		k =160		k =240		k =420	
	$\mathcal{P}$	δ	$\mathcal{P}$	δ	$\mathcal{P}$	δ	$\mathcal{P}$	δ	$\mathcal{P}$	δ
FedDLM	**573**	**0.29**	**401**	**0.38**	**237**	**0.46**	**193**	**0.59**	**143**	**0.69**
LIP	284	0.14	173	0.17	105	0.20	92	0.27	68	0.32
Sensitivity(block)	462	0.22	278	0.31	169	0.33	127	0.36	86	0.42

TABLE IV. VALIDATION OF THE PRECISION OF FEDDLM'S RISK LOCATION.

FL-based Classifier	k =40		k =80		k =160		k =240		k =420	
	γ	F1-score	γ	F1-score	γ	F1-score	γ	F1-score	γ	F1-score
FedIIoT	0.73	0.57	0.64	0.49	0.52	0.41	0.43	0.34	0.32	0.25
FedRAPID	0.82	0.61	0.75	0.55	0.66	0.48	0.54	0.40	0.39	0.29

VI. DISCUSSION

In the above experiments, we tested the effects of risk scoring and risk localization in different models and privacy parameters, proving the validity of our proposed leakage risk assessment framework FedDLM. Our proposed FedDLM can better capture the root cause of sensitive information leakage and its critical location in FL-based Android malware classifiers by comparing it with various baseline assessment methods. In the results of the experiment, we have the following inspirations for the leakage risk work and privacy defense work of the federal Android malware classifier:

To meter the degree of leakage risk of sensitive information, necessary to consider its distribution characteristics and essential leakage mechanisms; The coarse-grained model takes the degree of memory of all the information of the sample as the measure of risk, resulting in poor effect. It is because only by capturing the essence of sensitive information leakage at a fine-grained can effective and precision assessment results be obtained in different system settings (e.g., noise parameters, batch size, state of the model, etc.).

The class activation characteristics (for sensitive information) of the gradient in the Android malware classifier can provide practical operational recommendations for fine-grained and efficient privacy defense. For example, tighter privacy budgets are applied to gradient elements with a higher risk (higher class influence), while relaxed privacy budgets are applied to the remaining parts. It can alleviate the conflict between privacy and utility in the existing privacy protection mechanism.

VII. Conclusion

This paper proposes a fine-grained evaluation mechanism FedDLM for the risk of sensitive information leakage of FL-based Android classifiers according to the essential cause of leakage. FedDLM estimates the theoretical maximum inference ability of the attack from the perspective of information theory to meter the degree of leakage risk of the classifier effectively. It uses class activation features in the classifier to accurately identify critical locations in the shared gradient where leakage risks exist. The effectiveness and accuracy of FedDLM in assessing the risk of sensitive information disclosure are verified by simulation experiments on different models and privacy parameter Settings on the Androzoo dataset. The evaluation results provide valuable insights and practical defense suggestions for the information leakage problem of classifiers.

It should be emphasized that this paper estimates the opponent's maximum inference ability from the information content perspective. Estimation methods based on more accurate probabilities are beyond the scope of this paper and will be considered in future research work. In addition, we noticed a correlation between the degree of variation in the distribution of clients and the degree of breach risk. Therefore, we plan to explore the leakage risk assessment in non-IID and asynchronous settings.

Acknowledgment

This research was funded by the National Natural Science Foundation of China (No 62272024).

References

[1] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, “A survey of android malware detection with deep neural models,” ACM Computing Surveys (CSUR), vol. 53, no. 6, pp. 1–36, 2020.

[2] Z. Tu, H. Cao, E. Lagerspetz, Y. Fan, “Demographics of mobile app usage: Long-term analysis of mobile app usage,” CCF Transactions on Pervasive Computing and Interaction, vol. 3, pp. 235–252, 2021.

[3] R. G´alvez, V. Moonsamy, and C. Diaz, “Less is more: A privacy-respecting android malware classifier using federated learning,” arXiv preprint arXiv:2007.08319, 2020.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 2, pp. 1–19, 2019.

[5] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in 2019 IEEE symposium on security and privacy (SP), pp. 691–706, IEEE, 2019.

[6] ICO consultation on the draft AI auditing framework guidance for organisations ,2020, CrossRef.

[7] N. Singh, H. Kasyap, and S. Tripathy, “Collaborative learning based effective malware detection system,” in ECML PKDD workshop 2020, pp. 205–219, Springer, 2020.

[8] S. Shukla, P. S. Manoj, G. Kolhe, and S. Rafatirad, “On-device malware detection using performance-aware and robust collaborative learning,” in 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 967–972, IEEE, 2021.

[9] R. Taheri, M. Shojafar, M. Alazab, R. Tafazo, “Fed-iiot: A robust federated malware detection architecture in industrial iot,” IEEE transactions on industrial informatics, vol.17, no.12, pp. 8442–8452, 2020.

[10] V. Rey, P. M. S. S´anchez, A. H. Celdr´an, and G. Bovet, “Federated learning for malware detection in iot devices,” Computer Networks, vol. 204, p. 108693, 2022.

[11] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, “Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers,” International Journal of Security and Networks, vol. 10, no. 3, pp. 137–150, 2015.

[12] F. Tram`er, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Stealing machine learning models via prediction {APIs},” in 25th USENIX security symposium (USENIX Security 16), pp. 601–618, 2016.

[13] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in pharmacogenetics: An End-to-End case study of personalized warfarin dosing,” in 23rd USENIX security symposium (USENIX Security14), pp.17–32, 2014.

[14] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp. 1322–1333, 2015.

[15] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in 2017 IEEE symposium on security and privacy (SP), pp. 3–18, IEEE, 2017.

[16] 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 the IEEE/CVF conference on computer vision and pattern recognition, pp. 9311–9319, 2021.

[17] L. Song and P. Mittal, “Systematic evaluation of privacy risks of machine learning models,” in 30th USENIX Security Symposium (USENIX Security 21), pp. 2615–2632, 2021.

[18] A. Hannun, C. Guo, and L. Maaten, “Measuring data leakage in machine-learning models with IRFIL,” in arXiv 2102.11673,2021.

[19] S. Saeidian, G. Cervia, T. J. Oechtering, and M. Skoglund, “Quantifying membership privacy via information leakage,” IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3096–3108, 2021.

[20] D. Yu, G. Kamath, J. Kulkarni, J. Yin, T.-Y. Liu, and H. Zhang, “Per-instance privacy accounting for differentially private stochastic gradient descent,” arXiv preprint arXiv:2206.02617, 2022.

[21] P. Cheng, W. Hao, S. Dai, J. Liu, Z. Gan, and L. Carin, “Club: A contrastive log-ratio upper bound of mutual information,” in International conference on machine learning, pp. 1779–1788, PMLR, 2020.

[22] A. Badola, C. Roy, V. Padmanabhan, and R. P. Lal, “Decomposing the deep: finding class-specific filters in deep cnns,” Neural Computing and Applications, vol. 35, no. 18, pp. 13583–13596, 2023.

[23] M. Melis, M. Scalas, A. Demontis, D. Maiorca, B. Biggio, G. Giacinto, and F. Roli, “Do gradient-based explanations tell anything about adversarial robustness to android malware?,” International journal of machine learning and cybernetics, pp. 1–16, 2022.

[24] K. Allix, T. F. Bissyand´e, J. Klein, and Y. Le Traon, “Androzoo: Collecting millions of android apps for the research community,” in Proceedings of the 13th international conference on mining software repositories, pp. 468–471, 2016.

[25] A. Wainakh, F. Ventola, T. M¨uß, J. Keim, “User-level label leakage from gradients in federated learning,” arXiv preprint arXiv:2105.09369, 2021.

[26] B. Zhao, K. R. Mopuri, and H. Bilen, “idlg: Improved deep leakage from gradients,” arXiv preprint arXiv:2001.02610, 2020.

[27] C. Song and V. Shmatikov, “Overlearning reveals sensitive attributes,” arXiv preprint arXiv:1905.11742, 2019.

[28] J. Wang, S. Guo, X. Xie, and H. Qi, “Protect privacy from gradient leakage attack in federated learning,” in IEEE INFOCOM 2022-IEEE Conference on Computer Communications, pp. 580–589, IEEE,2019.

[29] J. Aalmoes, V. Duddu, and A. Boutet, “Dikaios: Privacy auditing of algorithmic fairness via attribute inference attacks,” arXiv preprint arXiv:2202.02242, 2022.

[30] X. Liu, H. Li, G. Xu, S. Liu, Z. Liu, and R. Lu, “PADL: privacy-aware and asynchronous deep learning for iot applications,” IEEE Internet Things J., vol. 7, no. 8, pp. 6955–6969, 2020.