GraphGuard: Detecting and Counteracting Training Data Misuse in Graph Neural Networks

Bang Wu*†[1], He Zhang†, Xiangwen Yang†, Shuo Wang*‡, Minhui Xue*, Shirui Pan§ and Xingliang Yuan†
*CSIRO's Data61, Australia
†Monash University, Australia
‡Shanghai Jiao Tong University, China
§Griffith University, Australia

***Abstract*—The emergence of Graph Neural Networks (GNNs) in graph data analysis and their deployment on Machine Learning as a Service platforms have raised critical concerns about data misuse during model training. This situation is further exacerbated due to the lack of transparency in local training processes, potentially leading to the unauthorized accumulation of large volumes of graph data, thereby infringing on the intellectual property rights of data owners. Existing methodologies often address either data misuse detection or mitigation, and are primarily designed for local GNN models rather than cloud-based MLaaS platforms. These limitations call for an effective and comprehensive solution that detects and mitigates data misuse without requiring exact training data while respecting the proprietary nature of such data. This paper introduces a pioneering approach called GraphGuard, to tackle these challenges. We propose a training-data-free method that not only detects graph data misuse but also mitigates its impact via targeted unlearning, all without relying on the original training data. Our innovative misuse detection technique employs membership inference with radioactive data, enhancing the distinguishability between member and non-member data distributions. For mitigation, we utilize synthetic graphs that emulate the characteristics previously learned by the target model, enabling effective unlearning even in the absence of exact graph data. We conduct comprehensive experiments utilizing four real-world graph datasets to demonstrate the efficacy of GraphGuard in both detection and unlearning. We show that GraphGuard attains a near-perfect detection rate of approximately 100% across these datasets with various GNN models. In addition, it performs unlearning by eliminating the impact of the unlearned graph with a marginal decrease in accuracy (less than 5%).**

I. Introduction

Graph Neural Networks (GNNs) have gained prominence in graph data analysis due to their exceptional performance and wide-ranging applications [12, 27, 32, 83], including e-Commerce, drug discovery, and protein folding [10, 18, 29, 30, 57, 58, 67, 69]. Currently, Machine Learning as a Service (MLaaS) platforms [4, 22, 26, 42] have revolutionized the deployment of GNN models by enabling model developers [35] to deploy them in the cloud and providing public APIs to end users. However, MLaaS platforms face transparency challenges, especially concerning models developed and trained locally. The lack of visibility in the local training process complicates monitoring for irregularities or illicit activities, raising concerns about potential *data misuse* during model training.

The *data misuse* problem during ML model training presents significant practical and security concerns. As deep learning models intrinsically rely on vast data sets for optimal functionality, model developers have a propensity to gather large amounts of data, sometimes through unauthorized means, occasionally or intentionally. Such misuse undermines the intellectual property rights of data owners. For example, recent ML tool developers, who deploy their ML models via MLaaS platforms like OpenAI and LensaAI, have been sued by authors and artists for unauthorized utilization of their intellectual content [19, 39]. These lawsuits have prompted policy development and legislation, with the EU introducing policies to prevent the misuse of training data [38].

The data misuse issue also gains prominence in the context of GNNs since they are utilized in sensitive domain applications, such as using GNNs to predict properties in Protein-Protein Interaction (PPI) graphs in pharmaceutical research [33, 34, 43, 47]. In such scenarios, nodes signify proteins and edges indicate their interactions [36, 59]. The construction of these graphs demands extensive experimentation and significant financial investments, making them valuable intellectual property [40, 44]. Many pharmaceutical firms, as GNN model developers, are seeking to leverage GNN to facilitate drug discovery, disease prevention, and diagnosis [40, 70, 72]. These companies might exploit data without owners' consent, seeking commercial gains through MLaaS predictions, thus bypassing the costs of data acquisition. Such unauthorized use severely breaches the data owner's IP rights. Therefore, there is a pressing need to protect graph data against potential data misuse.

Existing Studies. Existing methods are primarily designed for local deployment rather than cloud deployment of GNN models, rendering them less compatible with MLaaS platforms. In addition, they usually focus only on detecting or mitigating data misuse. *1) Detecting data misuse.* One common approach to determine whether specific graph data were used during model training is membership inference [17, 25, 37, 62]. However, they typically require analyzing outputs from the target model using original target samples, which might be less practical for detecting data misuse in MLaaS. Specifically,

Network and Distributed Systems Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.24441
www.ndss-symposium.org

[1] This work was partially done when Bang Wu was at Monash University.

TABLE I: Comparison between GraphGuard (our design) and Related Works. ○ indicates "Not Considered", ◐ indicates "Partially Considered", and ● indicates "Fully Considered".

Methods		Scope	Requirements			
		GNNs	R1 - Detectable	R2 - Remedial	R3 - Data Privatization	R4 - Model Agnostic
Passive MIA	[9], [73], [71], [15]	○	●	○	○	●
	[46], [62], [25], [17]	●	●	○	○	●
Active MIA	[50], [55]	○	●	○	○	◐
Unlearning-(SISA-based)	[6]	○	○	●	○	○
	[11]	●	○	●	○	○
Unlearning-(Others)	[8], [60], [16], [31], [82]	○	○	●	◐	●
	[14], [13], [64], [48]	●	○	●	○	◐
GraphGuard (Our Design)		●	●	●	●	●

data owners have to transfer their exact training samples to the cloud for membership inference, where they may be hesitant to do so due to the sensitivity of their graph data, intellectual property considerations, and data usage agreements. *2) Mitigating data misuse.* Most existing studies concentrate on mitigating data misuse through unlearning [6, 11, 14, 64], which involves removing the influence of certain training samples from the trained model. However, they call for particular function blocks in their GNN architecture or GNN training process to ensure unlearning, which is incompatible with general GNN models deployed on MLaaS platforms. Additionally, they also require the server to store or have access to the training graph for unlearning purposes, making them not feasible given the data owners' intellectual property considerations on their data transfer or storage on the cloud server.

Our Proposal. In this paper, we introduce an integrated pipeline, GraphGuard, to shield graph data from potential misuse in GNNs deployed via MLaaS. We start by formalizing the problem of data misuse in GNNs under MLaaS and listing the requirements for effective implementation in the context of MLaaS. Next, we propose a comprehensive pipeline that satisfies these requirements, including considerations of *misuse detection* (**R1**) and *mitigation* (**R2**), *data privatization* (**R3**), and *GNN model agnostic* (**R4**) (refer to Tab. I). The GraphGuard includes two main modules: proactive misuse detection and training-graph-free unlearning. The former safeguards data during detection, eliminating the need to transfer graph structures or model parameters among entities to protect their data privatization. The latter achieves mitigation also by considering such data privatization, and adding no extra assumptions on the training GNN model architecture.

Technical Challenges. Meeting all four requirements simultaneously is a non-trivial task. For example, in striving to fulfill both **R1** and **R3**, we observe that the efficacy of misuse detection techniques, such as membership inference (**R1**), could be significantly reduced when avoiding the transmission of graph structures or model parameters among entities (**R3**). Membership inference typically relies on analyzing how a model differentiates between member and non-member training data, which often hinges on overfitting. The overfitting effect is considerably reduced without access to the exact graph training data structure, thus attenuating the detection signal. Similarly, challenges arise in endeavoring to satisfy both **R2** and **R3**. The unlearning process (**R2**) necessitates the identification of specific samples for removal, and evaluating the impact of unlearning on the model is vital to guide this process. However, when the unlearned graph structure is not available (**R3**), gauging how the removal of samples influences the model becomes formidable, thus impeding the unlearning process.

Our Contributions. To meet requirements **R1**, **R3**, and **R4**, we present a novel proactive misuse detection module that detects graph data misuse without transmitting confidential graph structure data. It utilizes an alternative graph without structure during the misuse identification when analyzing the performance difference between misused/benign GNN models. To bolster such discernment distributions, we employ a radioactive graph construction method. When these graphs are used for training, the distribution difference of the predictions from the misused/benign GNN model is maximized, enhancing the misuse detection performance. We introduce a training-graph-free unlearning module to satisfy **R2**, **R3**, and **R4**. Our method uses fine-tuning with generated synthetic graphs to carry out unlearning, rather than depending on the exact unlearned graph. By using unlearning data samples for fine-tuning, the model increases the loss on these samples, effectively neutralizing their impact.

In summary, we make the following contributions.

- To the best of our knowledge, we present an innovative integrated pipeline framework, called **GraphGuard**, which is the first practical approach addressing the detection and mitigation of graph data misuse in GNNs within the MLaaS context.
- We define the problem of graph data misuse in MLaaS-deployed GNNs and identify four critical requirements for its mitigation: *detectable*, *remedial*, *data privatization*, and *model agnostic*.
- We introduce an integrated pipeline tailored to *GNNs in MLaaS*, which effectively addresses the outlined requirements. Our approach incorporates a novel misuse detection technique that leverages membership inference augmented with radioactive data, which strengthens the ability to discern between benign and misused model performance. Additionally, we propose an unlearning methodology that employs synthetic graphs to avoid using the confidential graph structure,

providing an effective means to mitigate the consequences of data misuse.

- We conduct extensive experiments on four real-world graph datasets and demonstrate the effectiveness of our design. Specifically, our design, GraphGuard, achieves an almost 100% detection rate across different GNN models, and facilitates unlearning with under 5% accuracy loss. GraphGuard is open-source and available at this repository: https://github.com/GraphGuard/GraphGuard-Proactive.

II. Preliminaries

This section introduces concepts, notations, and the application scenario of our paper, including GNNs, MLaaS, and membership inference in MLaaS.

A. Graph Neural Networks

Node Classification via GNNs. GNNs have shown great success in graph analysis tasks. This paper considers the node classification task. Formally, an attributed graph can be denoted as $G = (A, X)$, where $A \in \{0,1\}^{|V|\times|V|}$ indicates the graph structure, and $X \in \mathbb{R}^{|V|\times d}$ denotes the node features. $V = \{v_1, v_2, ..., v_{|V|}\}$ represents a set of nodes, and d indicates the dimension of node features (e.g., for node v_i, $X_i \in \mathbb{R}^{1\times d}$ denotes its features with dimension d). Given a set of nodes $V_t \subseteq V$ labeled with Y_t, a node classification GNN model f aims to label nodes based on both the graph structure A and node features X in G, i.e., $f : V \to Y$.

Generally, GNN for node classification has two different learning settings: *transductive* and *inductive*. **(1)** In the transductive setting, the inference graph of the GNN f is the same as the training graph, i.e., all nodes for inference have already been observed at the training time of f. **(2)** For the inductive setting, the inference graph of f is different from the training graph. This setting is similar to traditional learning settings where f learns the knowledge from the training graphs and is used to predict the node label in the unseen graphs.

GNN Model Formulation. There are various types of GNN architectures. For example, in a graph convolutional network (GCN) model [32], each layer aggregates the feature/embedding of both nodes and their neighbors by considering the graph structure. Formally, a 2-layer GCN model f is represented as:

$$f(A, X) = Softmax(\hat{A}ReLU(\hat{A}XW^{(0)})W^{(1)}), \quad (1)$$

where $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ represents the normalized adjacency matrix. Given $\tilde{A} = A + I_{|V|}$, $\tilde{D}$ indicates the diagonal degree matrix of $\tilde{A}$ (i.e., $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$). $W^{(0)}$ and $W^{(1)}$ represent the trainable parameters of f. $ReLU(\cdot)$ and $Softmax(\cdot)$ denote non-linear activation functions.

B. Machine Learning as a Service

MLaaS. MLaaS is trending as an emerging cloud service to train and serve machine learning models [4, 5, 22, 26, 42]. It enables model developers to build (*Training Service*) and deploy (*Serving Service*) their models and learning applications to model users conveniently and automatically. A notable framework is Amazon SageMaker [4], which has integrated a popular graph learning tool (i.e., DGL [54]) for the development and deployment of GNNs [53].

Model developers, who want to train a GNN model on their graph data, can first schedule (create and configure) a training job through the training service provided by MLaaS platforms. After that, the training service will receive the training job and launch a training instance that loads the training graph provided by the owners, and then train a GNN f accordingly. Finally, the well-trained f can be downloaded or saved to the cloud storage bucket for later use. To utilize the serving service, model developers first upload their locally trained GNN f to cloud storage or specify the location of an online GNN f, then the Hosting Service can create the API endpoints by utilizing the GNN f and its service configuration. These endpoints will remain alive, and model users can then obtain instantaneous predictions by querying these endpoints.

Inductive GNNs in MLaaS. Note that, in the inductive setting, the graphs used for the training and inference of GNNs are different. Therefore, MLaaS provides APIs for users to query their inference graph during serving. Specifically, the model developer uploads only a GNN f to the cloud when creating serving instances. GNN users upload their inference graphs and node IDs for predictions during the serving period. The serving endpoints then perform prediction on the inference graph and respond to the corresponding labels of the node IDs, respectively.

C. Membership Inference in MLaaS

Node-level Membership Inference. As a deep learning model, GNN models hunger for large amounts of data to obtain better performance, making the model developer need to gather as much data as possible, raising concerns about data misuse. A popular approach to identifying the training data of a given GNN f is membership inference (MI) [17, 25]. In the context of inductive node classification, given a target GNN f_{θ^*} trained on $G_m = (A_m, X_m)$, whether a node v_i is in the training dataset of f_{θ^*} (i.e., $\texttt{membership} = 0$ or 1) can be inferred by

$$\texttt{membership} = \begin{cases} 1 & \text{if } \texttt{mcs} > \eta, \\ 0 & \text{if } \texttt{mcs} \leq \eta, \end{cases} \texttt{mcs} = \mathcal{A}(f_{\theta^*}(G)_{v_i}), \quad (2)$$

where η indicates a threshold, and $\texttt{mcs}$ represents the membership confidence score. $f_{\theta^*}(G)_{v_i}$ denotes the prediction vector of node v_i on the queried graph G, and $\mathcal{A}(\cdot)$ represents an MI model whose output lies in the range $[0, 1]$.

In practical inference, since the initiator of MI can not access the training and non-training data of the target GNN f, they generally utilize a shadow dataset and shadow model to guide $\mathcal{A}$'s training. Specifically, to perform membership inference on the target GNN f_{θ^*} trained on G_m, attackers will first gather a shadow dataset $G' = G'_m \cup G'_n$ that shares the same domain as G_m. Then, they can generate a shadow GNN $f'_{\theta'}$ trained on G'_m, followed by obtaining the attack model $\mathcal{A}$ by solving $\max_{\mathcal{A}} d(\mathcal{A}(f'_{\theta'}(G'_m)), \mathcal{A}(f'_{\theta'}(G'_n)))$, where $\theta' = \arg\min_\theta L(f'_\theta(G'_m))$ and $d(\cdot,\cdot)$ indicates a distance function. As a result, a well-trained attack model $\mathcal{A}$ predicts 1 on member data (i.e., $\mathcal{A}(f'_{\theta'}(G'_m)) = 1$) while outputs 0 on non-member data (i.e., $\mathcal{A}(f'_{\theta'}(G'_n)) = 0$).

TABLE II: Capabilities of Different Entities in Our Application. ● represents full access, ◐ represents access without authorization, while ○ indicates no access. "Model Developer" denotes an adversarial model developer who gains access to A_p and X_p without the data owner's authorization.

Entity	G_p		G_m^0	f_{θ^*}
	A_p	X_p		
Data Owner	●	●	○	○
MLaaS Server	○	●	○	●
Model Developer	◐	◐	●	●

III. Problem Formulation

In this section, we focus on introducing the application settings and scenarios of our problem and the risk of data misuse in GNNs deployed via MLaaS. We first describe the system model by introducing the entities and their roles and goals. Then, we present the risk of data misuse in our scenarios. And finally, we present the objective for our design to mitigate this misuse problem.

A. System Model

In this paper, we consider a GNN model developed locally but deployed on an MLaaS server. We consider node classification as a basic and popular task for GNNs. We focus only on GNNs in inductive settings, which is more common for the model developer to use MLaaS for GNN deployment and serve other GNN users. Our application scenario includes three distinct entities:

(1) *Data Owner:* Data owners possess graph data, often regarded as their intellectual property. They have full access (both knowledge and modification capabilities) to their graph data $G_p = (A_p, X_p)$. In our scenario, they only consider the graph structure A_p to be private information, while less sensitive data X_p can be provided to others if necessary (refer to Sec. III-B).

(2) *Model Developer:* The model developer comprises entities that develop and own a well-trained GNN f_{θ^*}, and deploy f_{θ^*} on MLaaS platforms, thus severing GNN users by providing them with API to query f_{θ^*} and charge users and obtain commercial benefit. Benign model developer will develop their GNN models (e.g., f_{θ^*}) by gathering and training their model on an authorized training graph G_m^0 (e.g., they buy the graph data from data owners to benefit their GNN training), thus they have full access to both f_{θ^*} and G_m^0.

(3) *MLaaS Server:* The MLaaS server signifies the entity (e.g., Amazon, Microsoft, or Google) providing MLaaS (e.g., cloud computing services) to model developers. They help model developers deploy their GNNs (e.g., f_{θ^*}), thus having full access to the model. In addition, they may also receive other less sensitive information, such as X_p if necessary.

We summarize their capabilities in Tab. II. In addition, for presentation purposes, we summarize the notation introduced here and in the following sections in Tab. III.

TABLE III: Notations and Explanations.

Notation	Explanation
G_p	Graph owned by data owner, $G_p = (A_p, X_p)$
$\hat{G}_p$	Queried graph, $\hat{G}_p = (\hat{A}_p, X_p)$ and $\hat{A}_p = I$
G_m	Training graph of the model developer
G_m^0	Authorized training graph ($G_p \cap G_m^0 = \emptyset$)
G_m^1	Unauthorized training graph ($G_p \cap G_m^1 \neq \emptyset$)
$\tilde{G}_p$	Synthetic unlearning graph
$\tilde{G}_r$	Synthetic remaining graph
A_p	Adjacency matrix of G_p
$\hat{A}_p$	Adjacency matrix of $\hat{G}_p$ ($\hat{A}_p = I$)
X_p	Node attributes of G_p and $\hat{G}_p$
X_m^0	Node attributes of G_m^0
Y_p	Labels of the nodes in G_p and $\hat{G}_p$
f_{θ^*}	GNN trained by model developer
$f_{\theta_0^*}$	GNN trained on authorized graph G_m^0
$f_{\theta_1^*}$	GNN trained on unauthorized graph G_m^1
$f_{\widetilde{\theta^*}}$	Unlearned GNN
f_{θ^p}	Pre-trained surrogate model
g	Graph generation model
L	Loss function during GNN training
$\mathcal{A}$	Membership inference attack model

B. Threat Model

Within the MLaaS ecosystem outlined above, a critical concern is the threat of data misuse during model training, which compromises the IPs of the data owners. To scope our discussion, we assume that the data owner is benign, since they have no incentive to undermine their own IPs. Similarly, we assume the MLaaS server to be trusty, especially considering that service providers like Amazon and Google have well-established reputations. Contrarily, we focus on a scenario in which a model developer is an attacker, acquiring and exploiting unauthorized graph data during their GNN training process, thereby infringing upon the IPs of the data owner.

Motivations of Data Misuse. In this paper, we consider the model developer, who develops a model trained on unauthorized data and deploys it via MLaaS, as an attacker. This model is trained using meticulously gathered training data without the data owner's consent to exhibit superior performance. Such proficiency allows the model developer to gain commercial benefits by selling predictions through MLaaS, while avoiding costs associated with procuring the training data. This kind of unauthorized data usage severely undermines the IP rights of the data owner. It is worth noting that such a data misuse problem is not rare in practice. OpenAI, for example, faced lawsuits from various authors who claimed that their data were integrated into ML models without permission [19]. Similarly, artists have raised concerns about LensaAI, a tool that is alleged to train models using their artwork to reproduce specific styles, contending that it infringes on their rights [39].

Attacker Capabilities in Data Misuse. Considering the data misuse attack mentioned earlier, we attribute two capabilities to the attacker: 1) Obtaining data from the data owner without authorization. This unauthorized access often stems from

lapses in data management or oversight by the data owner. For example, several medical companies have experienced data breaches due to inadequate data management or system misconfigurations [7, 20, 21]. 2) Deploy a model within MLaaS while concealing its illicit use of unauthorized data. Since the training process is executed locally by the model developer, the MLaaS server lacks visibility into their local training activities. This makes it challenging to determine whether a model was trained on unauthorized data.

Based on the above attack setting, we formally provide the definition of training graph data misuse for GNNs in MLaaS as follows:

Definition 1: [Data Misuse for GNNs in MLaaS] Consider a model developer who intends to develop a well-trained GNN model using an authorized graph G_m^0. In the case of *data misuse*, the model developer, in addition to G_m^0, intentionally or occasionally collects an unauthorized graph G_p, and uses $G_m^1 = G_m^0 \cup G_p$ as a training set to construct the GNN f_{θ^*} in a local setting. Subsequently, the model developer provides the trained model f_{θ^*} to the MLaaS server for deployment and the GNN model service, without disclosing their use of unauthorized data.

Remarks. *1) Node Classification.* In this paper, we focus primarily on node classification as it is a foundational task in GNN applications. Given that graph classification and link prediction also rely on node embeddings, our methods can be readily extended to these tasks. A detailed discussion on how our design can be broadened to include other GNN tasks can be found in Sec. VI.

2) Inductive Settings. We focus on attacks targeting inductive GNNs, where the training and inference graphs are distinct. This setting makes detecting graph misuse challenging due to the inaccessibility of the training process in the MLaaS setting. In transductive settings, the inference graph is the same as the training graph, allowing the benign server an immediate opportunity to determine if the inference graph is unauthorized and has been used for training without consent.

C. Design Requirements

This paper aims to develop a framework to identify and mitigate data misuse in the context of GNN in MLaaS. Our method should be suitable for the MLaaS scenario that meticulously addresses the settings of each participating entity. Specifically, our design requirements are as follows.

- **R1–Detectable:** Our method must facilitate the detection of data misuse, allowing the identification of unauthorized GNN models deployed in MLaaS that have been trained by misusing graph data without authorization, even when the training process of the data misuse is less transparent (i.e., locally performed by the malicious model developer).
- **R2–Remedial:** Our framework should enable the MLaaS server to initiate or coordinate the unlearning process in cases where data misuse is detected. This involves either performing necessary actions to remove the impact of unauthorized data usage or protecting the data owner's forgotten right.
- **R3–Data Privatization:** A critical aspect of our design in MLaaS settings is protecting the data privatization of every entity throughout our mitigation process. Our system has multiple entities, and they will have their own data that will not be shared with others. For example, the data owner's graph structure should not be exposed to other entities during the mitigation process due to the data privatization concern or data usage agreements. This means that the data owner is not expected to share their graph structure with the MLaaS server or the model developer. Meanwhile, the MLaaS server and model developer will not share the GNN model with the data owner, and model developer will not share their training graph data with the server (see their knowledge in Tab. II).
- **R4–Model Agnostic:** We do not make any assumptions or requirements for the three entities. Specifically, for the model developer, we do not assume any additional process during the training of the GNN model. For MLaaS, we do not require specified architectures for the deployment of GNN models. Namely, our design can be applied to GNNs with a general model architecture in the context of MLaaS.

Remark for Data Privatization. Here we specify the privatized data for each entity that we considered in this paper. For the model developer and MLaaS server, well-trained GNN should be considered privatized data and not provided to the data owner or general GNN user. In addition, the graph structure for the data owner's and the model developer's training graphs should also be considered privatized data and not be provided to any other entities. This is a common scenario in GNN systems and has been the subject of previous studies on GNN security [15, 24, 63, 71, 77]. Collecting training graphs and building a GNN model often requires a large amount of human, computing, and economic resources, and thus a model developer is concerned about the privatization of its training graphs and GNNs and would not provide them to others. Meanwhile, the data owner should keep the graph structure information privatized, and never be provided to other entities in our system.

Note that our design focuses only on the graph structure. Given that graph data primarily represent connections and interactions between node entities, the graph structure tends to contain more sensitive information in GNN applications [63]. A practical example could be a company managing a private transaction graph, where nodes represent individuals and node attributes (i.e., features) represent publicly available basic profiles. Edges represent sensitive transactions among individuals. These edges are considered the property of the company, and they may express concern over the privatization of their graph structure data. Another example could be a GNN used for PPI in drug discovery, where nodes represent proteins and node attributes represent well-known properties [36, 59]. The edges represent interactions among proteins, which should be obtained by extensive experimentation [40, 44]. We also discuss how our design can be extended to also protect the node attributes in Sec. VI.

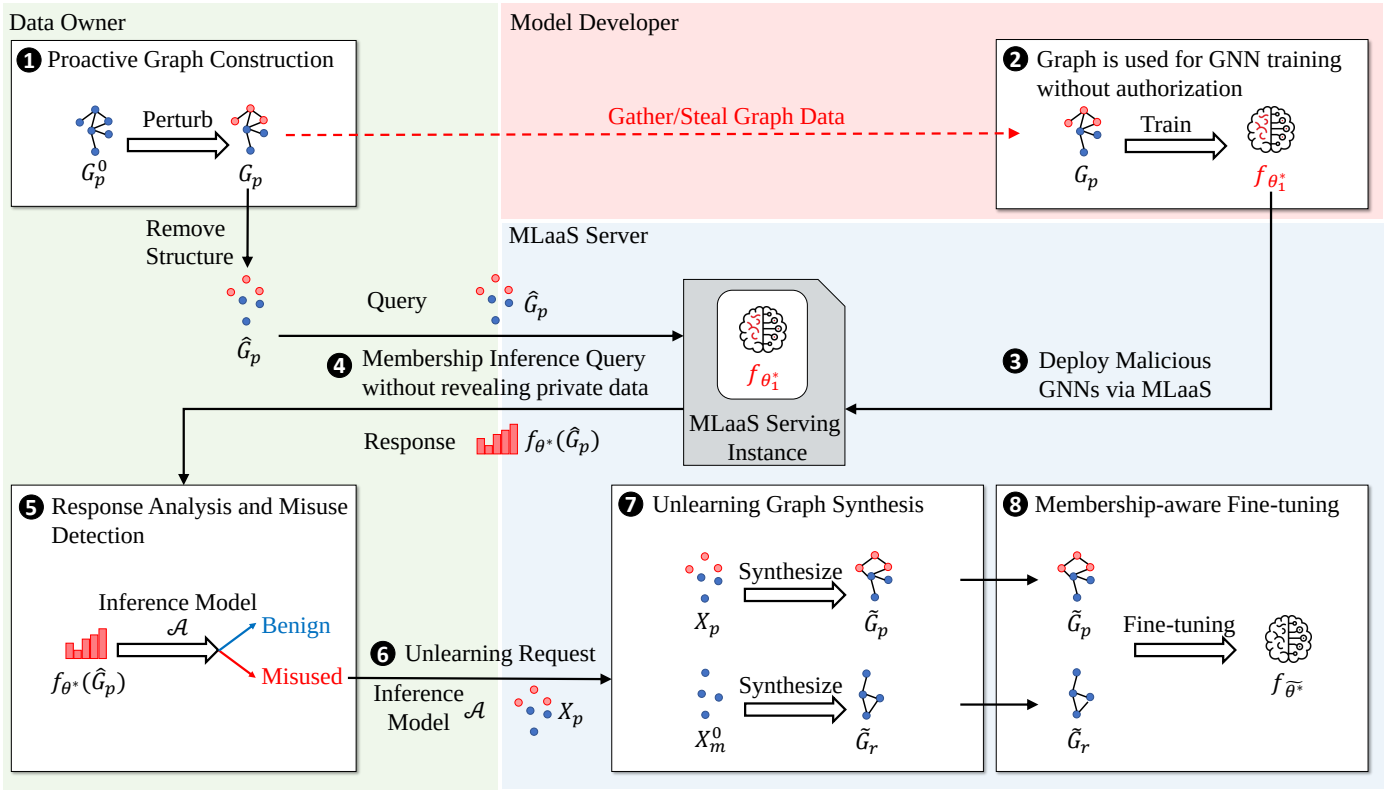

Fig. 1: Overall Pipeline of Our Graph Data Misuse Mitigation Framework.

IV. GraphGuard

To deal with the training graph misuse problem in GNNs, in this paper, we introduce an integrated framework, GraphGuard, which can both detect and mitigate the data misuse and is suitable in the context of MLaaS. Specifically, we split the data misuse mitigation into two stages of GraphGuard (i.e., detecting data misuse and reducing its effects) to match the requirements R1 and R2, followed by proposing two respective components, misuse detection and unlearning. Moreover, we make relevant adjustments or enhancements to the misuse detection and unlearning components to address other requirements (i.e., R3 and R4). A detailed flow chart of this process is presented in Fig. 1. In this section, we will introduce each of these two components by first defining their overall goals, technical challenges when considering the practical MLaaS scenarios, design intuition, and then the detailed methodology.

A. Graph Data Misuse Detection

Graph data misuse detection concentrates on determining whether misuse has occurred in the context of a suspect GNN model deployed on an MLaaS platform. Within our framework, data owners will perform this detection task. Since they do not have direct access to the suspect GNN model, they may issue queries to the MLaaS server and ask them to perform inference on these queries and generate a response to assist in their misuse detection. Formally, the objective of our detection module can be elucidated as follows:

Definition 2: [Graph Data Misuse Detection for GNNs in MLaaS] Assuming a suspect GNN f_{θ^*}, which is trained on a graph G_m, is now deployed in MLaaS. With only the ability to query f_{θ^*}, the data owner aims to identify if graph data $G_p = (A_p, X_p)$ was used in training f_{θ^*} (i.e., whether $G_m \cap G_p = \emptyset$) without divulging private information A_p.

It can be found that the above detection has a similar objective as the membership inference attack we introduced in Sec. II, but with additional constraints on preventing the disclosure of information regarding A_p. Therefore, we can consider our detection as a membership inference problem, while adding noise to the membership inference query samples to protect sensitive information in A_p. Namely, to prevent the private sample from being disclosed, rather than querying precise samples $G_p = (A_p, X_p)$, our membership inference process can query a perturbed version of G_p to the target model deployed on the server.

Challenges and Design Intuitions. However, introducing noise to the query samples makes membership inference considerably less effective. Note that membership inferences are based on the analysis of the responses to the query sample [25]. Specifically, whether a specific sample has been used for training is determined based on whether the target model overfits the query samples [9, 71]. Therefore, when the query samples are perturbed, even if they are training members, the target model will exhibit reduced overfitting on them, making them harder to accurately classify as members.

To address this challenge, we draw inspiration from the concept of radioactive data, as first introduced in [50]. Instead of relying solely on overfitting, we advocate for constructing

a *radioactive graph* to facilitate the detection of graph misuse. Specifically, we introduce optimized perturbations to the original graph data, yielding radioactive graphs. We expect that any model trained using our radioactive graph will bear a unique mark. This mark can render the output distributions of a vanilla model and a data-misused model trained on the radioactive graph distinctly different.

Design Overview. By utilizing the radioactive graph for membership inference, our data misuse detection process will comprise two phases, including an initialization phase for the radioactive graph construction, and a detection phase for membership inference, which can be succinctly outlined as follows:

Initialization Phase:

1. The data owner converts their original data into a radioactive graph locally via our radioactive graph construction algorithm.

After the radioactive graph is generated and stored, the potential misuse occurs as how we discussed in Sec. III-B. Specifically, the radioactive graph may be illicitly acquired and utilized by adversarial model developers to augment their models without the data owner's consent. These developers then deploy their model to MLaaS platforms and market their prediction services.

When the data owner believes that a model on MLaaS might be exploiting their data for training (e.g., due to its task relevance), they initiate the detection phase.

Detection Phase:

2. The data owner crafts a query about their radioactive graph, while ensuring the omission of structural details (i.e., removing all edges, querying only individual nodes coupled with their attributes).

3. The MLaaS performs inference on the suspected GNN model with the query graph and subsequently transmits the results to the data owner.

4. The data owner evaluates the received response to discern if the operational model on MLaaS was trained using the radioactive graph data without the data user's authorization.

Radioactive Graph Construction. With the aforementioned workflow in place, our goal is to construct the radioactive graph. We first define the optimization problem of radioactive graph construction and then present how to solve it.

Our radioactive graph is designed to render the output distributions of the vanilla model and data misused model difference. A common strategy for assessing such differences is to follow similar strategies as membership inference and calculate the prediction disparity D of these outputs. Specifically, as shown in Sec. II-C, the distribution difference for member and non-member data can be calculated as: $d(\mathcal{A}(f_{\theta^*}(G_m)), \mathcal{A}(f_{\theta^*}(G_n)))$. Accordingly, in the context of increasing the difference by the radioactive graph, the objective

Algorithm 1 Radioactive Graph Construction

Input:
$G_p^0 = (A_p, X_p^0)$ Initial graph owned by the data owner,
Y_p Labels of the graph owned by the data owner,
N Maximum construction epoch,
k Feature construction step,
$f_{\theta^p} = f_{\theta^p}^c \circ f_{\theta^p}^e$ A pretrained GNNs.
Output:
$G_p = (A_p, X_p)$ Radioactive graph.

for n **from** 0 **to** $N-1$ **do**
  $G_p^{n+1} = (A_p, X_p^{n+1}) \leftarrow G_p^n = (A_p, X_p^n)$
  $L_{opt} \leftarrow ||f_{\theta^p}^e(G_p^n) - f_{\theta^p}^e(\hat{G}_p^n)||_2 - L(f_{\theta^p}(\hat{G}_p^n), Y_p)$
  $\Delta_L \leftarrow \frac{\partial L_{opt}}{\partial X_p^0}$
  $X_p^{'} \leftarrow$ Selecting the top-k largest $X_p[i]$ w.r.t. $|\Delta_L[i]|$
  for $X_p[i] \in X_p^{'}$ **do**
    if $\Delta_L[i] > 0$ **then**
      $X_p^{n+1}[i] \leftarrow 0$
    else
      $X_p^{n+1}[i] \leftarrow 1$

of the output distribution can be represented as:

$$\begin{aligned} &\max_{G_p} d(\mathcal{A}(f_{\theta_1^*}(\hat{G}_p)), \mathcal{A}(f_{\theta_0^*}(\hat{G}_p))), \\ &s.t.\ \theta_1^* = \arg\min_{\theta} L(f_\theta(G_m^1)), \\ &\theta_0^* = \arg\min_{\theta} L(f_\theta(G_m^0)), \end{aligned} \tag{3}$$

where $\mathcal{A}(\cdot)$ is a membership inference attack model, $d(\cdot,\cdot)$ indicates a distance function, $\hat{G}_p = (\hat{A}_p, X_p) = (I_{|V|}, X_p)$ is the queried graph without graph structure information (i.e., all nodes in $\hat{G}_p$ are isolated), G_m^1 and G_m^0 represent two different versions of G_m, namely, G_m^1 indicates the G_m that (partially) includes G_p (i.e., $G_p \cap G_m^1 \neq \emptyset$) while G_m^0 denotes the G_m that excludes G_p (i.e., $G_p \cap G_m^0 = \emptyset$).

Note that, obtaining $\hat{G}_p$ from (3) is non-trivial and (3) contains unspecified functions such as $\mathcal{A}(\cdot)$ and $d(\cdot,\cdot)$. Therefore, we convert the optimization problem in (3) as follows:

$$\min_{X_p} \ ||f_{\theta^p}^e(G_p) - f_{\theta^p}^e(\hat{G}_p)||_2 - L(f_{\theta^p}(\hat{G}_p), Y_p), \tag{4}$$

where L is GNN's training loss function, $f_{\theta^p} = f_{\theta^p}^c \circ f_{\theta^p}^e$ is a pre-trained surrogate model (e.g., any publicly available GNN model performing similar tasks, which is consistent with prior works' assumption on graph unlearning [11]). $f_{\theta^p}^e$ and $f_{\theta^p}^c$ indicate the classifier and encoder of f_{θ^p}, respectively. $\circ$ represents the composition of functions. This optimization can be efficiently solved using the gradient descent method. We provide the proof of the above conversion in App. -A. A step-by-step procedure for constructing the radioactive graph is delineated in Algorithm 1.

Discussion. *a) Privacy Risk of Querying Node Attribute.* A potential concern relates to link stealing attacks, which can leverage node prediction results to deduce the graph structure. Such querying and responding could potentially introduce additional vulnerability to link leakage [24, 63]. However, it should be noted that all existing link-stealing attacks [24, 63] require inference results on a graph that includes a private link.

On the contrary, our approach involves inference on a graph without a private link. Additionally, we introduce perturbations into the node attributes, thereby reducing the effectiveness of link recovery methods that use graph structure learning.

b) Distinction from Backdoor and Poisoning Approaches. Other works have proposed the injection of data into the training dataset [66, 80, 81]. However, our approach has distinct settings and objectives. Existing approaches assume that an attacker can influence the training process, while the data owner remains unaware if their data have been used during training. Our goal, on the other hand, is to amplify the difference between member and non-member data, while other approaches aim to cause misclassification or reduce model performance.

c) Distinction from Radioactive Data. Other work [50] also proposed to perturb data for detection. However, they focus only on linear models and Euclidean data. Our design considers GNNs which are nonlinear models and used for analyzing graph structure data. Their design is not applicable to our case since 1) their detection heavily relies on the assumption that the model is linear, while common GNN has nonlinear activation layers; 2) their design focus on attribute and ignore the graph structure information.

d) Distinction from Model Watermarking. Other studies have suggested embedding watermarks into the ML model during its training phase. Note that these techniques serve distinct objectives compared to our design: they aim to protect the IP of the model developer, not that of the data owner. As a result, their settings differ from ours. Specifically, watermarking is implemented by the model developer, who often has full access to the training period. In contrast, in our context, the data misuse detection is carried out by the data owner, who does not have access to the training process.

B. Graph Data Misuse Mitigation

This component focuses on the implementation of unlearning in the infringing GNN model to eliminate the influence of unauthorized training data samples. The MLaaS server will undertake this task and perform unlearning after receiving a request from the data owner regarding a deployed infringing GNN model. Since the MLaaS servers can access and modify the GNN model via retraining or fine-tuning, but do not have access to the training graph data, they may ask the data owner, who issued the unlearning request, to provide less sensitive information (i.e., node attributes) to facilitate the unlearning process. Specifically, we formalize the objective of our second component, i.e., graph unlearning, as follows:

Definition 3: [Training-graph-free Unlearning for GNNs in MLaaS] Given that G_m^0 indicates the authorized graph, a GNN f_{θ^*} is identified as data-misused GNN $f_{\theta_1^*}$ when it is trained in G_m^1, where $G_m^1 = G_m^0 \cup G_p$, and $G_p = (X_p, A_p)$ represents an unauthorized graph. Without accessing the sensitive graph structure of G_p and the authorized graph G_m^0 (i.e., A_p and A_m^0), the MLaaS server aims to reconstruct an unlearned GNN $f_{\widetilde{\theta^*}}$, approximating the benign GNN $f_{\theta_0^*}$ trained in G_m^0.

Challenges and Design Intuitions. In the above mitigation process, a key stipulation is that, the graph structure, as the private information, must remain undisclosed to the server. This presents significant challenges compared to previous unlearning studies. In particular, most existing processes [11, 13, 14, 64] require that the entities performing the unlearning can access the unlearning samples so that these samples guide which information to forget. In the absence of these samples, the server remains clueless about which specific knowledge should be expunged from the GNN model undergoing unlearning, and making ordinary unlearning methods inapplicable.

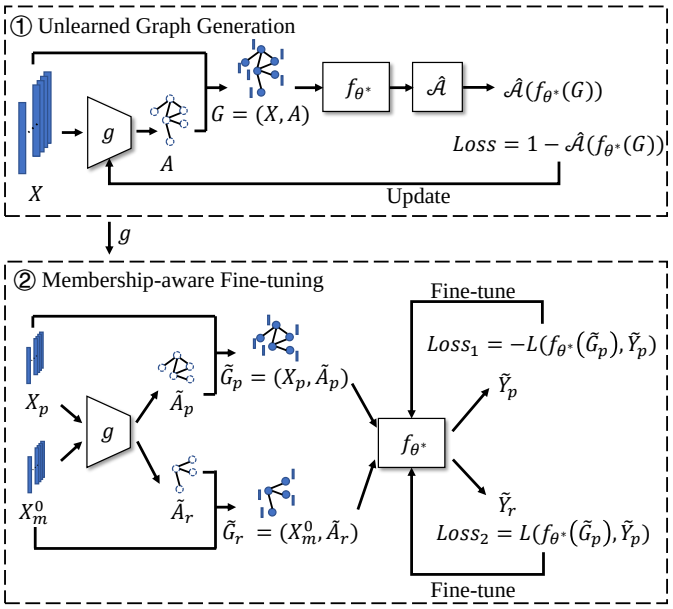

Fig. 2: Overall Architecture of Training-graph-free Unlearning.

To deal with this challenge, without dispatching the unlearning samples directly to the server, we propose executing the unlearning with fine-tuning based on synthetic graphs self-generated by the server. The graph synthesis is based on statistical knowledge derived from a membership inference attack model, augmented by some auxiliary knowledge (for instance, less sensitive node attributes supplied by the data owner). This approach eschews the direct sharing of sensitive graph structure data. Nonetheless, the unlearning remains directed by the information embodied in the membership inference model. Note that the membership inference model encapsulates the knowledge the target model has acquired, which precisely contains what we intend the target model to forget.

Detailed Methodologies. Building upon the above intuitions, our mitigation strategy unfolds in two distinct stages: the generation of the unlearned graph and membership-aware fine-tuning. Specifically, our unlearning module consists of two steps: unlearned graph generation and membership-aware fine-tuning. The overall architecture of our proposed solution is illustrated in Fig. 2.

a) Unlearned Graph Generation. Given access to less-sensitive node attributes, the MLaaS server will request the data owner to upload the X_p (i.e., node attributes of the nodes to be unlearned) upon receiving an unlearning request. With these attributes at its disposal, which can be viewed as isolated nodes, the server will construct their interconnections. An intuitive design would be to directly connect nodes with similar

node attributes due to the homophily of graph data. However, this approach falls short as it only considers the knowledge of the node attribute, while ignoring the original connection between these nodes. Consequently, performing unlearning on these generated graphs may lead to inadequate unlearning.

To this end, we propose to utilize the knowledge from the target GNN f_{θ^*}. Concretely, we train a model $g = g^c \circ g^e : X \to (A, X)$ to connect nodes and then generate a graph $G = (A, X)$, where g^c and g^e indicate the classifier and encoder of g, respectively. Given access to the node features X, the inference model $\mathcal{A}$ and the target GNN f_{θ^*}, g is trained by the following optimization.

$$\min_{g} 1 - \mathcal{A}(f_{\theta^*}(g(X))), \tag{5}$$

where an edge exists between two nodes v_i and v_j (i.e., $A_{i,j} = 1$) when $||g^e(v_i) - g^e(v_j)||_2 \leq \epsilon_u$, ϵ_u indicates a predefined threshold. The intuition of Equation (5) is that, the inference model $\mathcal{A}$ outputs 1 if the generated graph $g(X)$ is overfitted by the target unlearning model f_{θ^*}. Note that, in addition to the node features X_p of the unlearning nodes, we also request the model developer to provide X_m^0 (i.e., node features of the authorized training graph) and involve it in the training of g, which facilitates the following fine-tuning to maintain the performance of f_{θ^*}. With the well-trained g, we obtain the unlearning graph $\tilde{G}_p = g(X_p)$ and the remaining graph $\tilde{G}_r = g(X_r)$ for the following membership-aware fine-tuning.

b) Membership-aware Fine-tuning. The objective of unlearning is to make the target model f_{θ^*} forget the unlearned graph samples while maintaining performance on the remaining graph samples. In this paper, we proposed using the fine-tuning methodology to conduct GNN unlearning, which is motivated by the fact that a GNN model learns by updating its weights to minimize training loss in its training graph. Specifically, to maintain performance when unlearning, the unlearned GNN $f_{\widetilde{\theta^*}}$ is obtained by

$$\widetilde{\theta^*} = \arg\min_{\theta} = L(f_\theta(\tilde{G}_r)) - \alpha L(f_\theta(\tilde{G}_p)), \tag{6}$$

where L represents the training loss function, α indicates a pre-defined hyper-parameter to balance the unlearning request and GNN utility. Training loss is calculated by comparing the prediction results with the initial prediction labels generated by f_{θ^*} before fine-tuning. Note that α can be adjusted based on the size ratio between $\tilde{G}_r$ and $\tilde{G}_p$. For example, a smaller unlearning graph $\tilde{G}_p$ might require lower values of α. In our design, we use 0.5 to balance the utility of the model and the effectiveness of unlearning.

Remarks. The generation of our unlearned graph does not necessitate the exact reconstruction of private edges. Instead, both the unlearning and remaining graph data are synthesized exclusively from node attributes, removing edge information (see Fig. 2 and Sec. IV-B). And our unlearned graph generation model also does not rely on the private graph structure, but utilizes latent patterns deriving from the inference attack model and node attributes (see Equation (5) and Sec. IV-B).

C. Combination

As shown in Fig. 1, we propose our graph misuse mitigation framework by combining the *proactive misuse detection* (i.e., Sec. IV-A) and *training-graph-free unlearning* (i.e., Sec. IV-B) together. Specifically, our framework mainly includes two phases, as shown in the following.

TABLE IV: GNN Training Graph Statistics.

Datasets	# Nodes	# Edges	# Attributes
Cora	2,708	5,429	1,433
Citeseer	3,327	4,732	3,703
Pubmed	19,717	44,338	500
Flickr	89,250	899,756	500

1) The data owner will first construct the proactive graph G_p by perturbing the node attributes of the benign graph G_p^0 by our radioactive graph construction algorithm (described in Sec. IV-A).

2) Data misuse occurs, where model developer may misuse G_p intentionally or occasionally where $f_{\theta^*} = f_{\theta_1^*}$ is trained on $G_m^1 = G_p \cup G_m^0$ (as described in Sec. III-B).

3) The model developer deploy the data-misused model $f_{\theta_1^*}$ via MLaaS server.

4) When data owners' concerns on data misuse of a suspect GNN $f_{\theta_1^*}$ arise, they will submit membership inference queries on $\hat{G}_p$ (whose structure has been removed for privacy consideration) to the MLaaS server. The MLaaS server processes the inference employing the targeted GNN $f_{\theta_1^*}$, subsequently relaying the prediction results from $f_{\theta_1^*}(\hat{G}_p)$ to the data owner.

5) Data owners, based on the confidence scores of these query results (i.e., $f_{\theta_1^*}(\hat{G}_p)$), determine whether data misuse occurred.

6) Once identifying data misuse The data owner will request the MLaaS server to perform unlearning by providing their node attributes X_p and inference model $\mathcal{A}$ for their unlearned graph.

7) The MLaaS server will also request the GNN model developer to provide the node attributes from their authorized training graph X_m^0. And the MLaaS server generates the unlearned graph $\tilde{G}_p$ based on X_p, and the remaining graph $\tilde{G}_r$ based on X_m^0 (as our method in Sec. IV-B).

8) The MLaaS server can finally obtain an unlearned GNN $f_{\widetilde{\theta^*}}$ based on both $\tilde{G}_p$ and $\tilde{G}_r$ by following our unlearning method (as described in Sec. IV-B).

Among the above, Steps 1, 4, and 5 are our *proactive misuse detection* module; Steps 2 and 3 are the data misuse threat; and Steps 6 to 8 are our *training-graph-free unlearning* module.

V. Experiments and Evaluations

A. Experiment Setup

Evaluation Settings. In the design of our graph data misuse mitigation framework, we have focused on four design requirements as shown in Sec. III-C.

R1 - Misuse Detection Effectiveness. To evaluate how our design can effectively detect misuse, we define a successful

TABLE V: Comparison of AUC between Our Method and Baseline Membership Inference.

	GCN			GraphSage			GAT			GIN		
	Baseline	Ours	Δ	Baseline	Ours	Δ	Baseline	Ours	Δ	Baseline	Ours	Δ
Cora	0.874	0.999	↑**0.125**	0.864	0.999	↑**0.135**	0.927	1.0	↑**0.073**	0.857	1.0	↑**0.143**
Citeseer	0.711	0.999	↑**0.288**	0.822	1.0	↑**0.178**	0.723	0.999	↑**0.276**	0.767	1.0	↑**0.233**
Pubmed	0.906	1.0	↑**0.094**	0.902	1.0	↑**0.098**	1.0	1.0	**0**	0.932	1.0	↑**0.068**
Flickr	1.0	1.0	**0**	0.994	1.0	↑**0.006**	0.996	1.0	↑**0.004**	0.998	1.0	↑**0.002**

TABLE VI: Comparison of TPRs (%) under low FPRs (1%) between Our Method and Baseline Membership Inference.

	GCN			GraphSage			GAT			GIN		
	Baseline	Ours	Δ	Baseline	Ours	Δ	Baseline	Ours	Δ	Baseline	Ours	Δ
Cora	0.257	0.945	↑**0.688**	0.596	1.0	↑**0.404**	0.202	0.953	↑**0.751**	0.174	0.740	↑**0.566**
Citeseer	0.832	1.0	↑**0.168**	0.881	1.0	↑**0.119**	0.897	1.0	↑**0.103**	0.815	1.0	↑**0.185**
Pubmed	0.618	1.0	↑**0.382**	0.360	1.0	↑**0.640**	0.717	1.0	↑**0.283**	0.407	1.0	↑**0.593**
Flickr	0.997	1.0	↑**0.003**	0.966	1.0	↑**0.034**	0.998	1.0	↑**0.002**	0.998	1.0	↑**0.002**

detection by the case when the data owner correctly identifies a node that is/is not used during the training. Since such misuse detection is a binary classification problem, we use the detection rate and calculate the corresponding *AUC*. We evaluate the results among all nodes in both authorized and unauthorized graphs.

R2 - Unlearning Effectiveness. To evaluate how our design can effectively eliminate the impacts of unlearned nodes, we follow similar evaluation metrics to those of unlearning. Specifically, we use the *attack successful rates* of the membership inference attack on both the unlearned and remaining nodes to evaluate the effectiveness of unlearning. The *attack successful rate* of the membership inference attack in GNN is defined by the MIA attacker correctly identifying a node that is/is not used during training. We evaluate the results among all nodes in both authorized and unauthorized graphs.

R3 - Data Privatization. We have shown that our design satisfied R3 in our previous sections. That is, the data owner, model developer, and MLaaS server will not directly provide their data to other entities. Specifically, the data owner queries only isolated nodes without graph structure. The model developer will also not provide his training graph structure to the MLaaS server during the unlearning process. The MLaaS server will also not provide direct access to the deployed GNN model to the data owner's request for misuse detection.

R4 - GNN Model Agnostic. We have shown that our design satisfies R4 since it does not require any specific GNN architectures as in our previous sections. In this section, we will also show that our design is generally suitable for different popular and common GNN architectures and datasets. In addition, we further evaluate the robustness of our misuse detection method by assuming that the model developer uses the graph denoising technique to update our radioactive graph data.

Datasets. Our design is evaluated on four widely recognized public graph datasets, namely, Cora [32], Citeseer [32], Pubmed [32], and Flickr [74], all of which have been previously employed in GNN security analysis. A summary of the dataset statistics can be found in Tab. IV. Cora, Citeseer, and Pubmed are citation networks where nodes signify publications, and node attributes denote the presence of specific keywords. An edge between nodes signifies a connection between two publications. Flickr is an image dataset, with nodes representing individual images and node attributes detailing the image profiles. An edge between nodes indicates common information between images, such as being originating from the same location or being submitted to the same gallery.

GNN Models. We evaluate our design on four state-of-the-art GNN models: GCN [32], GraphSage [23], Graph Attention Networks (GATs) [56], and GIN [68]. The number of features in the hidden layer for all GNN models is 16. The activation function for the hidden layer is ReLU. We use the Adam optimizer with a learning rate of 0.01 and training epochs of 300. The loss function of our model is the cross-entropy loss.

MLaaS Settings. For ethical considerations, we do not test our methods on a real MLaaS. Instead, to simulate the MLaaS setting, we restrict data access for each entity to resemble the environment inherent to MLaaS (see Tab. II). Furthermore, we emulate data transmission, ensuring that all transmitted data are the same as the communications in the actual MLaaS and that no private graph structure is transmitted through the system.

B. Evaluating Proactive Misuse Detection - (R1, R4)

This section evaluates the effectiveness of our proactive misuse detection module on four datasets and four GNN models by comparing membership inference results and distribution differences between member and non-member data.

Effectiveness. Tab. V shows the AUC of our detection method. It can be found that, for all four GNN models and datasets, our design can achieve almost 1 AUC. Namely, we can find a threshold to perfectly identify whether a node has been used during target model training. While our detection shows that it is effective among different datasets and GNN models, we further show that our design is general, which not only does not have an assumption on the training process and model architecture, but is also effective for different domain tasks and GNN models.

To demonstrate the effectiveness of our design in detecting graph misuse, we take into account the practical requirements

Fig. 3: Comparison of Output Distribution from Misused (in orange colors)/Benign (in blue colors) GNNs between Our method (left) and Baseline method (right).

for low false positive rates (FPRs) during detection. We evaluate TPRs (correctly detect misuse) with low FPRs (false alarms of detection). The results are presented in Tab. VI. Across the four GNN models and datasets, our design consistently achieves much higher TPRs (i.e., almost 100%) than the baseline method with low FPRs, i.e., only 1%.

Output Distribution Comparisons. To further demonstrate the effectiveness of our radioactive graph construction in amplifying the performance disparity between benign and misuse GNN models, we compare the output distributions from both types of models in Fig. 3. Note that, in accordance with requirement R3, all these distributions are derived using $\hat{G}_p$, which safeguards the confidentiality of the graph structure. The results reveal that, without using our radioactive graph, the output distributions of benign and misuse models substantially overlap. This overlap arises because GNNs naturally learn from and potentially overfit to the graph structure. Consequently, the differences in output when querying $\hat{G}_p$, which does not contain graph structure information, are therefore negligible. However, with the introduction of our radioactive graph, the data-misused GNNs tend to learn more from the attributes, resulting in a marked distinction in performance compared to benign models. Based on the figures, there are only a few overlaps between the authorized and unauthorized nodes. Thus, it is convenient for the data owner to identify whether their graphs have been misused by the model developer.

Comparison with Baseline. We consider the baseline following the strategies in [46]. Note that their method requires the data owner to query the exact authorized graph or have direct access to the suspect GNN model to obtain accurate predictions of the nodes. This method is more suitable to be applied locally, which does not satisfy R3 in our MLaaS settings. Tab. V compares our method with the baseline design. It can be found that, even with a stronger assumption on the knowledge of the inference graph data, our design can still achieve better performance than their method.

Robustness of Our Design. We further discuss the robustness of our design. In practice, model developers may use graph denoise methods to preprocess the training graph they gathered. This could happen occasionally since using these methods to improve the model performance is common. This could also be done by the advanced model developer who is aware of the proactive method that injects noise to the graph similar to us, and proposes to use the denoise method to remove these proactive perturbations.

Tab. VII shows the AUC of the misuse detection with and without the denoising method in [28]. It could be found that, in all datasets and the GNN model, the AUC with the denoising method is almost the same as the AUC without the denoising method. Actually, this is because the graph denoise method commonly removes or adds the graph structure. However, our proactive design forces the GNN model to learn more from the attributes. Thus, common methods that denoising graphs by modifying the graph structure can not remove our injected proactive attributes. Namely, our design can be robust to these

graph denoise methods.

C. Evaluating Training-graph-free Unlearning - (R2, R4)

In this section, we evaluate the effectiveness of our unlearning design on four datasets and four GNN models.

Effectiveness. To validate the effectiveness of our unlearning design, we assess our unlearning method and perform experiments on three datasets and four GNN models. Tab. IX illustrates the membership inference attack's accuracy of the different datasets and GNN models. It can be found that our design can significantly reduce the MIA attack success rate to about 0.5 attack accuracy. Namely, it is a random guess of the membership of a target node (since MI is a binary classification problem). Meanwhile, the attack successful rate for MIAs targeting the model before unlearning is much higher (e.g., about 80%). Considering that MIA assesses how the model remembers training samples, our design can significantly eliminate the impact of these unlearning nodes.

Utility. Additionally, we evaluate the utility of our unlearning method. Tab. VIII presents the accuracy of the GNN both prior to and after the unlearning process. It is evident that our approach incurs only a marginal decrease in the accuracy of the model, within 5%. This minor reduction in accuracy is reasonable, as the unlearning process involves the removal of some knowledge embedded in the unlearned samples. Consequently, trained GNNs possess less information from the training graph, which naturally leads to a slight decrease in the accuracy of the model.

Time Cost. We compare the efficiency of our method with training from scratch that retrains the GNN in the remaining training set without unlearned samples. As shown in Tab. X, we observe an efficiency improvement on all four GNN models, We observed that the relative efficiency improvement of smaller datasets (Cora and Citeseer) is more than that of larger dataset (Pubmed). For instance, the unlearning time improvement is about 3-6× for both Cora, Citeseer dataset, and about 1-2× for the Pubmed dataset. This is because a larger amount of time is required to train a larger graph from scratch. Although our method can significantly reduce the training efforts of the target model, it introduces new time costs for graph synthesis, which is also related to the size of the graph. However, we want to note that such computation cost is done by the MLaaS server which naturally contains more computation resources. Thus, our design can still be suitable for MLaaS settings.

VI. Discussion

Practicality in MLaaS Scenarios. Our design pipeline is well-suited for real-world MLaaS systems for two primary reasons: 1) Both proactive detection and training-graph unlearning only rely on using fundamental functions (e.g., training instance and prediction API) within MLaaS [1, 2, 41]. The detection performed by the data owner only queried the prediction APIs of the MLaaS server, which are basic functions provided by the MLaaS serving service [2, 41]. Similarly, the unlearning process executed by the server uses standard ML training and inference functions to generate unlearned graphs and unlearn the target model, which are available on general MLaaS platforms [1, 3, 41]. 2) Most of the computational overhead is borne by the server side. In our design, processes such as the training of the graph generation model or the fine-tuning of the target model are all handled by the MLaaS server, which possesses enough computational resources for these tasks [1, 3, 41].

Generalization to Other Tasks. Our design currently targets the node classification task in GNNs, but the underlying principles can be generalized to other tasks within GNNs. In nongraph-structured domains, similar strategies have been applied to image data and linear classifiers in [50] (we compare our design with theirs in Sec. IV-A). In other GNN tasks (e.g., graph classification and link prediction), our design can also be lent to them. Typically, the results for graph classification and link prediction are based on node embeddings, similar to those in node classification [65]. Given that our design aims to distinguish the output distributions derived from node embeddings between benign and data-misused GNNs, we can also vary distribution outcomes for both graph classification and link prediction using our strategies. We consider adaptation to other GNN tasks for future exploration.

Generalization to Protecting the Node Attributes. Currently, our design focuses on protecting the graph structure. However, our approach can also be expanded to protect node attributes. Specifically, regarding misuse detection, our current strategy is to construct proactive graphs that remain effective for membership inference even when their graph structure undergoes perturbation (i.e., all edges are removed). To protect the node attribute, we can further update our current proactive graph construction algorithm to consider both the graph structure and the node attributes of the proactive graph to be perturbed (i.e., updating X_p to $G_p = (A_p, X_p)$ in Equation (4)). Similarly, in the context of unlearning, our current methodology synthesizes only the unlearning training graph structure based on given node attributes. We can refine this by evolving our graph structure generation algorithm into one that simultaneously creates both the graph structure and node attributes. We envision this as a future step in our research.

Malicious Server. In this paper, our primary focus is on malicious model developers. Note that our design retains its capability to detect data misuse, even when the MLaaS server is malicious. Specifically, our detection method relies on querying the MLaaS server's prediction APIs (accessible to ordinary model users [1, 2, 41]). Once the server responds to these queries, the subsequent stages of the detection process are conducted locally on the data owner's side. Therefore, even in scenarios where the server rejects unlearning requests, our design remains effective in identifying data misuse.

VII. Related Work

Membership Inference. Existing studies demonstrates that machine learning models are vulnerable to attacks [49, 61, 75, 76, 78, 79]. For privacy attacks on GNNs [77], according to the inference manner, existing membership inference attacks can be categorized into passive inference methods and active inference methods. Passive MIAs [9], [73], [71], [15] aim to detect membership without elaborate active efforts (e.g., modifying data), while attackers in active MIAs [50], [55] will introduce elaborate efforts to facilitate detection. However, it is not easy to directly adapt these studies on independent

TABLE VII: Comparison of AUC with/without Denoising Mechanism [28].

	GCN			GraphSage			GAT			GIN		
	W	W/o	Δ	W	W/o	Δ	W	W/o	Δ	W	W/o	Δ
Cora	1.0	0.999	↓**0.001**	1.0	0.999	↓**0.001**	1.0	1.0	**0**	1.0	1.0	**0**
Citeseer	1.0	0.999	↓**0.001**	1.0	1.0	**0**	1.0	0.999	↓**0.001**	1.0	1.0	**0**
Pubmed	1.0	1.0	**0**	1.0	1.0	**0**	1.0	1.0	**0**	1.0	1.0	**0**
Flickr	1.0	1.0	**0**	1.0	1.0	**0**	1.0	1.0	**0**	1.0	1.0	**0**

TABLE VIII: Comparison of Model Accuracy (% is omitted) between the Unlearned Model (denoted as U) and Retrained Model (denoted as R).

	GCN			GraphSage			GAT			GIN		
	R	U	Δ	R	U	Δ	R	U	Δ	R	U	Δ
Cora	75.7	74.3	↓ **1.2**	67.4	66.5	↓ **0.9**	83.1	81.5	↓ **1.6**	86.4	85.1	↓ **1.3**
Citeseer	81.1	80.0	↓ **1.1**	70.0	68.7	↓ **1.3**	82.2	80.1	↓ **2.1**	79.5	78.9	↓ **0.6**
Pubmed	81.8	79.8	↓ **2.0**	82.5	80.3	↓ **2.2**	83.6	81.3	↓ **2.3**	83.6	82.8	↓ **0.8**

TABLE IX: Comparison of MIA Successful Rate(% is omitted) of the Unlearned Nodes Before/After Unlearning.

	GCN			GraphSage			GAT			GIN		
	Before	After	Δ	Before	After	Δ	Before	After	Δ	Before	After	Δ
Cora	86.9	51.8	↓ **35.1**	83.3	54.5	↓ **28.8**	85.6	47.5	↓ **38.1**	91.7	47.9	↓ **43.8**
Citeseer	91.3	68.7	↓ **22.6**	81.2	56.1	↓ **25.1**	61.4	60.3	↓ **1.10**	86.2	46.2	↓ **40.0**
Pubmed	93.6	49.2	↓ **44.4**	85.7	53.2	↓ **32.5**	82.4	49.7	↓ **32.7**	84.1	47.6	↓ **36.5**

TABLE X: Comparison of Time Cost (in seconds) for Retraining the Model (denoted as R) and Our Unlearning Method.

	GCN			GraphSage			GAT			GIN		
	R	Ours	Times(↑)	R	Ours	Times(↑)	R	Ours	Times(↑)	R	Ours	Times(↑)
Cora	3.615	0.725	≈**4.99**	4.188	0.643	≈**6.51**	3.600	0.720	≈**5.0**	4.26	1.225	≈**3.48**
Citeseer	1.746	0.375	≈**4.66**	2.023	0.333	≈**6.08**	1.737	0.375	≈**4.63**	2.058	0.613	≈**3.56**
Pubmed	4.201	3.043	≈**1.38**	4.865	2.670	≈**1.82**	4.190	3.017	≈**1.39**	4.968	5.124	≈**0.97**

and identically distributed (IID) data to graph data, where connections between nodes break the IID assumption in these works.

To this end, Olatunji et al. [46] first introduce MIAs in GNNs. Their approach was an extension of the general MIA methodologies in DNNs applied to GNNs. They employ a learning-based model to analyze the output confidence scores of target nodes' predictions and to ascertain their membership. On the other hand, Wu et al. [62] applied MIAs to GNNs for graph classification. Their attack methods relied on the prediction probability vector and the metrics computed based on it, focusing primarily on graph-level GNN models. In contrast, He et al. [25] concentrated on MIAs against node-level classification GNNs. They considered the membership of a target node by evaluating the confidence scores for node prediction. This evaluation involved the target nodes and their immediate (0-hop) and secondary (2-hop) neighbors. Conti et al. [17] further studied label-only MIA against GNNs specifically for the node classification task. They hypothesized that attackers could obtain only node labels.

It is worth noting that all these MIAs require attackers to query the target model with the target graph data. This target graph comprises both node attributes and the graph structure, enabling a comprehensive analysis of the response, regardless of whether it is confidence scores or labels only. However, these attacks differ from our practical settings without directly querying the private target graph in MLaaS.

Machine Unlearning. Based on the technology used for unlearning, current machine unlearning works include SISA-based methods [6] and others [8], [60], [16], [31], [82]. In SISA-based methods [6], the training dataset is partitioned into multiple shards and then used to train multiple sub-models individually, followed by integrating these submodels together to serve model users. Once unlearning requests are received, the model developer can only retrain one specific submodel to achieve fewer unlearning efforts [45]. Unlike SISA-based methods, other designs [45], such as modifying model parameters with influence functions, can also be used in unlearning. More details can be found in recent machine unlearning surveys [45, 52].

Since its inception by Cao et al. [8], the concept of machine unlearning has generated a wealth of strategies for its implementation in GNNs. Among them, Chen et al. [11] pioneered an approach named Graph Eraser, employing the Shard, Isolate, Sanitize, and Aggregate (SISA) method. They divided the training graph data into shards using graph partitioning, training each independently. Upon receiving an unlearning request, the model provider would retain the relevant

shard model, trained on the subgraph that encompasses the unlearning data. Despite its effectiveness, it was dependent on an ensemble architecture and required retraining of the shard model using training graph data.

Chien et al. [14] developed the first certified graph unlearning framework for GNNs, providing theoretical unlearning guarantees. However, their methodology was only applicable to linear GNNs. Similarly, Cheng et al. [13] introduced GNNDelete, an unlearning strategy that involved adding extra weight matrices to negate the effect of unlearning data. Despite its ingenuity, the design required an additional block where the unlearning data was stored, reducing its practicality in the MLaaS setting. Wu et al. [64] devised an unlearning strategy, defining an influence function known as the Graph Influence Function (GIF), which is considered an additional loss term for influenced neighbors, facilitating the unlearning of the graph structure. Moreover, Pan et al. [48] suggested an unlearning technique for GNNs that employed the Graph Scattering Transform, providing provable performance guarantees. However, both methods share a common requirement: precise unlearning samples must be utilized during the unlearning process to accurately eliminate the influence of specific data samples on the target unlearn model.

VIII. Conclusion

In this paper, we introduce *GraphGuard*, a pioneering integrated pipeline designed to address the graph data misuse issue in the context of GNNs deployed through MLaaS. Initially, we formalize the problem of graph data misuse concerning GNNs in the MLaaS setting and outline four critical requirements that reflect the practical considerations of MLaaS. Our proposed pipeline comprises two key components: a proactive graph misuse detection module, which adeptly detects graph misuse while respecting data privatization, and a training-graph-free unlearning module, which is capable of performing unlearning without access to training graph data and remains versatile across common GNN models. Through extensive experiments on four real-world graph datasets employing four state-of-the-art GNN models, we demonstrate the high effectiveness of GraphGuard.

Acknowledgements

We would like to thank Neil Gong and Xu Yuan for their insightful discussions and feedback at the initial stage of this work and thank the anonymous shepherd and reviewers for their helpful comments and feedback. This work is supported in part by a Monash-Data61 Collaborative Research Project (CRP43) and Australian Research Council (ARC) DP240103068, FT210100097, and DP240101547. Minhui Xue, Xingliang Yuan and Shirui Pan are also supported by CSIRO – National Science Foundation (US) AI Research Collaboration Program.

References

[1] [Online]. Available: https://d1.awsstatic.com/events/Summits/awstorontosummit/Scaling_your_ISV_AI_ML_offerings_using_Amazon_Sage_Maker_ISV201.pdf

[2] [Online]. Available: https://cloud.google.com/architecture/ml-on-gcp-best-practices

[3] [Online]. Available: https://cloud.google.com/architecture/guidelines-for-developing-high-quality-ml-solutions

[4] Amazon Web Services. (Accessed 2023) Amazon sagemaker - machine learning platform. [Online]. Available: https://aws.amazon.com/pm/sagemaker/

[5] AWS Labs, "GraphStorm," https://github.com/awslabs/graphstorm, accessed on June 29, 2023.

[6] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *IEEE Symposium on Security and Privacy*. IEEE, 2021, pp. 141–159.

[7] A. Burky, "Advocate aurora says 3m patients' health data possibly exposed through tracking technologies," Oct 2022. [Online]. Available: https://www.fiercehealthcare.com/health-tech/advocate-aurora-health-data-breach-revealed-pixels-protected-health-information-3

[8] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2015, pp. 463–480.

[9] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr, "Membership inference attacks from first principles," in *IEEE Symposium on Security and Privacy*. IEEE, 2022, pp. 1897–1914.

[10] L. Chen, Y. Liu, Z. Zheng, and P. S. Yu, "Heterogeneous neural attentive factorization machine for rating prediction," in *CIKM 2018*. ACM, pp. 833–842.

[11] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "Graph unlearning," in *CCS*. ACM, 2022, pp. 499–513.

[12] Y. Chen, L. Wu, and M. J. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," in *NeurIPS 2020,*.

[13] J. Cheng, G. Dasoulas, H. He, C. Agarwal, and M. Zitnik, "GNNDelete: A general strategy for unlearning in graph neural networks," in *ICLR*. OpenReview.net, 2023.

[14] E. Chien, C. Pan, and O. Milenkovic, "Certified graph unlearning," in *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

[15] C. A. Choquette-Choo, F. Tramèr, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 1964–1974.

[16] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. S. Kankanhalli, "Zero-shot machine unlearning," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 2345–2354, 2023.

[17] M. Conti, J. Li, S. Picek, and J. Xu, "Label-only membership inference attack against node-level graph neural networks," in *AISec@CCS*. ACM, 2022, pp. 1–12.

[18] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *AAAI 2021*. AAAI Press, pp. 4027–4035.

[19] S. Frenkel and S. A. Thompson, ""not for machines to harvest": Data revolts break out against a.i." Jul 2023. [Online]. Available: https://www.nytimes.com/2023/07/15/technology/artificial-intelligence-models-chat-data.html?searchResultPosition=1

[20] S. Gatlan, "Healthcare giant chs reports first data breach in goanywhere hacks," Mar 2023. [Online]. Available: https://www.bleepingcomputer.com/news/security/healthcare-giant-chs-reports-first-data-breach-in-goanywhere-hacks/

[21] D. Geer, "Medical informatics engineering breach: The gift that keeps on giving," Jan 2019. [Online]. Available: https://medium.com/the-aftermath-of-a-data-breach/medical-informatics-engineering-breach-the-gift-that-keeps-on-giving-9948231d2e95
[22] Google Cloud. (Accessed 2023) Google cloud vertex ai. [Online]. Available: https://cloud.google.com/vertex-ai
[23] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.
[24] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks." in *USENIX Security Symposium*, 2021, pp. 2669–2686.
[25] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, "Node-level membership inference attacks against graph neural networks," *CoRR*, vol. abs/2102.05429, 2021.
[26] IBM. (Accessed 2023) Ibm watson machine learning for z/os. [Online]. Available: https://www.ibm.com/docs/en/wml-for-zos
[27] D. Jin, L. Wang, H. Zhang, Y. Zheng, W. Ding, F. Xia, and S. Pan, "A survey on fairness-aware recommender systems," *Inf. Fusion*, vol. 100, p. 101906, 2023.
[28] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *KDD*. ACM, 2020, pp. 66–74.
[29] W. Jin, J. M. Stokes, R. T. Eastman, Z. Itkin, A. V. Zakharov, J. J. Collins, T. S. Jaakkola, and R. Barzilay, "Deep learning identifies synergistic drug combinations for treating covid-19," *Proceedings of the National Academy of Sciences*, vol. 118, no. 39, p. e2105070118, 2021.
[30] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.
[31] J. Kim and S. S. Woo, "Efficient two-stage model retraining for machine unlearning," in *CVPR Workshops*. IEEE, 2022, pp. 4360–4368.
[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR (Poster)*. OpenReview.net, 2017.
[33] C. Kooli and H. Al Muftah, "Artificial intelligence in healthcare: a comprehensive review of its ethical concerns," *Technological Sustainability*, 2022.
[34] M. Kop, "Ai & intellectual property: Towards an articulated public domain," *Tex. Intell. Prop. LJ*, vol. 28, p. 297, 2019.
[35] X. Liu, B. Wu, X. Yuan, and X. Yi, "Leia: A lightweight cryptographic neural network inference system at the edge," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 237–252, 2022.
[36] Y. Liu, H. Yuan, L. Cai, and S. Ji, "Deep learning of high-order interactions for protein interface prediction," in *KDD*. ACM, 2020, pp. 679–687.
[37] Z. Liu, X. Zhang, C. Chen, S. Lin, and J. Li, "Membership inference attacks against robust graph neural network," in *CSS*, ser. Lecture Notes in Computer Science, vol. 13547. Springer, 2022, pp. 259–273.
[38] H. A. K. LLP, "European parliament agrees on position on the ai act," Jun 2023. [Online]. Available: https://www.huntonprivacyblog.com/2023/06/15/european-parliament-agrees-on-position-on-the-ai-act/
[39] S. E.-D. Mattei, "Artists voice concerns over the signatures in viral lensaai portraits," Dec 2022. [Online]. Available: https://www.artnews.com/art-news/news/signatures-lensa-ai-portraits-1234649633/
[40] R. Melo, R. Fieldhouse, A. Melo, J. D. Correia, M. N. D. Cordeiro, Z. H. Gümüş, J. Costa, A. M. Bonvin, and I. S. Moreira, "A machine learning approach for hot-spot detection at protein-protein interfaces," *International journal of molecular sciences*, vol. 17, no. 8, p. 1215, 2016.
[41] Microsoft, "Deploy machine learning models to online endpoints for inference - azure machine learning." [Online]. Available: https://learn.microsoft.com/en-us/azure/machine-learning/how-to-deploy-online-endpoints?view=azureml-api-2&tabs=azure-cli
[42] Microsoft Azure. (Accessed 2023) Azure machine learning. [Online]. Available: https://azure.microsoft.com/en-au/products/machine-learning
[43] J. B. Mitchell, "Artificial intelligence in pharmaceutical research and development," pp. 1529–1531, 2018.
[44] R. Mrowka, A. Patzak, and H. Herzel, "Is there a bias in proteome research?" *Genome research*, vol. 11, no. 12, pp. 1971–1973, 2001.
[45] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," *CoRR*, vol. abs/2209.02299, 2022.
[46] I. E. Olatunji, W. Nejdl, and M. Khosla, "Membership inference attack on graph neural networks," in *TPS-ISA*. IEEE, 2021, pp. 11–20.
[47] A. Oliva, S. Grassi, G. Vetrugno, R. Rossi, G. Della Morte, V. Pinchi, and M. Caputo, "Management of medico-legal risks in digital health era: a scoping review," *Frontiers in medicine*, vol. 8, p. 2956, 2022.
[48] C. Pan, E. Chien, and O. Milenkovic, "Unlearning graph classifiers with limited data resources," in *WWW*. ACM, 2023, pp. 716–726.
[49] Y. Qin, J. Hu, and B. Wu, "Toward evaluating the robustness of deep learning based rain removal algorithm in autonomous driving," in *SecTL@AsiaCCS*. ACM, 2023, pp. 1–7.
[50] A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou, "Radioactive data: tracing through training," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 8326–8335.
[51] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *NeurIPS*, 2018, pp. 6106–6116.
[52] T. B. Shaik, X. Tao, H. Xie, L. Li, X. Zhu, and Q. Li, "Exploring the landscape of machine unlearning: A comprehensive survey and taxonomy," *CoRR*, vol. abs/2305.06360, 2023.
[53] J. Simon, "Now available on amazon sagemaker: The deep graph library," Dec 2019. [Online]. Available: https://aws.amazon.com/blogs/aws/now-available-on-amazon-sagemaker-the-deep-graph-library/
[54] Themeix, Jul 2018. [Online]. Available: https://www.dgl.ai/pages/about.html
[55] F. Tramèr, R. Shokri, A. S. Joaquin, H. Le, M. Jagielski, S. Hong, and N. Carlini, "Truth serum: Poisoning machine learning models to reveal their secrets," in *CCS*.

ACM, 2022, pp. 2779–2792.
[56] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR (Poster)*. OpenReview.net, 2018.
[57] Y. Wan, Y. Liu, D. Wang, and Y. Wen, "GLAD-PAW: graph-based log anomaly detection by position aware weighted graph attention network," in *PAKDD 2021*, ser. Lecture Notes in Computer Science, vol. 12712. Springer, pp. 66–77.
[58] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-scale commodity embedding for e-commerce recommendation in alibaba," in *KDD 2018*. ACM, pp. 839–848.
[59] X.-W. Wang, L. Madeddu, K. Spirohn, L. Martini, A. Fazzone, L. Becchetti, T. P. Wytock, I. A. Kovács, O. M. Balogh, B. Benczik *et al.*, "Assessment of community efforts to advance network-based prediction of protein–protein interactions," *Nature Communications*, vol. 14, no. 1, p. 1582, 2023.
[60] A. Warnecke, L. Pirch, C. Wressnegger, and K. Rieck, "Machine unlearning of features and labels," in *NDSS*. The Internet Society, 2023.
[61] B. Wu, S. Wang, X. Yuan, C. Wang, C. Rudolph, and X. Yang, "Defeating misclassification attacks against transfer learning," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 2, pp. 886–901, 2023.
[62] B. Wu, X. Yang, S. Pan, and X. Yuan, "Adapting membership inference attacks to GNN for graph classification: Approaches and implications," in *ICDM*. IEEE, 2021, pp. 1421–1426.
[63] F. Wu, Y. Long, C. Zhang, and B. Li, "Linkteller: Recovering private edges from graph neural networks via influence analysis," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2005–2024.
[64] J. Wu, Y. Yang, Y. Qian, Y. Sui, X. Wang, and X. He, "GIF: A general graph unlearning strategy via influence function," in *WWW*. ACM, 2023, pp. 651–661.
[65] L. Wu, P. Cui, J. Pei, and L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore, 2022.
[66] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," *CoRR*, vol. abs/2006.11890, 2020.
[67] F. Xie, L. Chen, Y. Ye, Z. Zheng, and X. Lin, "Factorization machine based service recommendation on heterogeneous information networks," in *ICWS 2018*. IEEE, pp. 115–122.
[68] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*. OpenReview.net, 2019.
[69] H. Yang, "Aligraph: A comprehensive graph neural network platform," in *KDD*. ACM, 2019, pp. 3165–3166.
[70] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea *et al.*, "Analyzing learned molecular representations for property prediction," *Journal of chemical information and modeling*, vol. 59, no. 8, pp. 3370–3388, 2019.
[71] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri, "Enhanced membership inference attacks against machine learning models," in *CCS*. ACM, 2022, pp. 3093–3106.
[72] Z. You, M. Zhou, X. Luo, and S. Li, "Highly efficient framework for predicting interactions between proteins," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 731–743, 2017.
[73] X. Yuan and L. Zhang, "Membership inference attacks and defenses in neural network pruning," in *USENIX Security Symposium*. USENIX Association, 2022, pp. 4561–4578.
[74] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. K. Prasanna, "Graphsaint: Graph sampling based inductive learning method," in *ICLR*. OpenReview.net, 2020.
[75] H. Zhang, B. Wu, S. Wang, X. Yang, M. Xue, S. Pan, and X. Yuan, "Demystifying uneven vulnerability of link stealing attacks against graph neural networks," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 41 737–41 752.
[76] H. Zhang, B. Wu, X. Yang, C. Zhou, S. Wang, X. Yuan, and S. Pan, "Projective ranking: A transferable evasion attack method on graph neural networks," in *CIKM*. ACM, 2021, pp. 3617–3621.
[77] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy graph neural networks: Aspects, methods and trends," *CoRR*, vol. abs/2205.07424, 2022.
[78] H. Zhang, X. Yuan, Q. V. H. Nguyen, and S. Pan, "On the interaction between node fairness and edge privacy in graph neural networks," *CoRR*, vol. abs/2301.12951, 2023.
[79] H. Zhang, X. Yuan, C. Zhou, and S. Pan, "Projective ranking-based GNN evasion attacks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8402–8416, 2023.
[80] S. Zhang, H. Chen, X. Sun, Y. Li, and G. Xu, "Unsupervised graph poisoning attack via contrastive loss back-propagation," in *WWW*. ACM, 2022, pp. 1322–1330.
[81] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in *SACMAT 2021*. ACM, pp. 15–26.
[82] Z. Zhang, Y. Zhou, X. Zhao, T. Che, and L. Lyu, "Prompt certified machine unlearning with randomized gradient smoothing and quantization," in *NeurIPS*, 2022.
[83] Y. Zheng, H. Zhang, V. C. Lee, Y. Zheng, X. Wang, and S. Pan, "Finding the missing-half: Graph complementary learning for homophily-prone and heterophily-prone graphs," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 42 492–42 505.

APPENDIX

A. Radioactive Graph Construction

Function-similar MI Model - Substituting $\mathcal{A}$. Since acquiring a well-trained MI model $\mathcal{A}$ may also be challenging for the data owner, our design employs a simplified but practical inference model to substitute the ideal $\mathcal{A}$. Note that the primary function of $\mathcal{A}$, whose value lies in the range $[0, 1]$, is to capture the overfitting of the target models, i.e., to ideally output 1 on members (i.e., G_m, training/overfitted samples) and 0 on non-members (i.e., G_n, testing/under fitted samples) of the target GNN f_{θ^*}. Instead of training $\mathcal{A}$ with the shadow dataset and model, as shown by the practical MI in Sec. II-C, we propose to substitute the well-trained $\mathcal{A}$ with a function-similar while practical indexing function $\hat{\mathcal{A}}$, which is underpinned by the intuition that the learning of $\mathcal{A}$ is to distinguish the difference of confidence scores between members and non-members of GNNs. Formally, $\hat{\mathcal{A}}$ is defined as

$$\hat{\mathcal{A}}(f_{\theta^*}(G)_{v_i}) = f_{\theta^*}(G)_{v_i}[y_i], \tag{7}$$

where $f_{\theta^*}(G)_{v_i}$ denotes the predicted labeling result on the node v_i from the input graph G, $[y_i]$ indicates the indexing function which selects the y_i-th element in vector $f_{\theta^*}(G)_{v_i}$, and y_i represents the ground-truth label of v_i. Note that, given Equation (7), $\hat{\mathcal{A}}$ denotes the prediction confidence score of the nodes in the query graph.

Incorporating $\hat{G}_p$ and $\hat{\mathcal{A}}$, the basic graph data misuse detection strategy in MLaaS is defined as follows: Given a suspect target GNN f_θ^* trained on G_m, which may have unauthorizedly used the target graph $G_p = (A_p, X_p)$ for training (i.e., $G_m \cap G_p \neq \emptyset$), the data owner of G_p can detect misuse by querying this target GNN f_{θ^*} with graph $\hat{G}_p = (\hat{A}_p, X_p)$, followed by calculating a membership confidence score $\texttt{mcs}$ for G_p:

$$\texttt{mcs} = \frac{1}{|V_{\hat{G}_p}|} \sum_{i=1}^{|V_{\hat{G}_p}|} \hat{\mathcal{A}}(f_{\theta^*}(\hat{G}_p)_{v_i}), \tag{8}$$

where $|V_{\hat{G}_p}|$ indicates the node number of $\hat{G}_p$. When $\texttt{msc}$ is close to 1, it is more likely that G_p was used during the training phase, while a $\texttt{msc}$ value closer to 0 indicates that it is less probable that G_p was used during the training of f_{θ^*}.

Objective Function Convertion. In this paper, from the view of the data owner, a desired G_p should satisfy $\mathcal{A}$ is expected to satisfy $\mathcal{A}(f_{\theta^*}(G_p)) = 1$ when θ^* is influenced by G_p during the training of f, otherwise $\mathcal{A}(f_{\theta^*}(G_p)) = 0$. Similarly, G_p and $\mathcal{A}$ are respectively substituted by $\hat{G}_p$ and $\hat{\mathcal{A}}$ to avoid exposing the sensitive graph structure and meet the practical capability of the data owner. Therefore, we propose that the proactive graph G_p can be obtained by solving the following optimization to help identify if G_p is used in the training of the target GNN f_{θ^*}:

$$\begin{aligned} &\min_{G_p} 1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p)) + \hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p)), \\ &s.t.\ \theta_1^* = \arg\min_\theta L(f_\theta(G_m^1)), \\ &\quad \theta_0^* = \arg\min_\theta L(f_\theta(G_m^0)), \end{aligned} \tag{9}$$

where G_m^1 and G_m^0 represent two different versions of G_m, namely, G_m^1 indicates the G_m which (partially) includes G_p (i.e., $G_p \cap G_m^1 \neq \emptyset$) while G_m^0 denotes the G_m which excludes G_p (i.e., $G_p \cap G_m^0 = \emptyset$). Note that the first element $1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$ targets the case G_m^1, while the second item $\hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$ is designed for the case G_m^0.

Optimization Solving. Obtaining $\hat{G}_p$ from (9) is non-trivial due to two reasons. **(a)** *Uncertain* $f_{\theta^*}(\hat{G}_p)$. Given that the data owner can only query f_{θ^*} in the context of MLaaS, the data owner cannot determine the source of the querying results (i.e., $f_{\theta_1^*}(\hat{G}_p)$ or $f_{\theta_0^*}(\hat{G}_p)$), resulting in a dilemma for the data owner when selecting the optimization goal (i.e., either $\min_{\hat{G}_p} 1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$ or $\min_{\hat{G}_p} \hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$). **(b)** *Either-or Optimization*. Assuming that the source of query results $f_{\theta^*}(\hat{G}_p)$ has been determined, only one item (i.e., $1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$ or $\hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$) can be involved in solving the whole optimization (9), which potentially contributes to sub-optimal $\hat{G}_p$.

To this end, we propose to devise customized solutions for different items in optimization 9, and then combine these solutions together with a novel *feature-loss-max* principle. **(1) Customized Solutions.** For items $1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$ and $\hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$, the customized solutions are listed below.

Case-A: $1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$. In addition to the above two reasons, another challenge in solving $\min_{G_p} 1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$ is the interdependence between G_p and θ_1^*. On the one hand, θ_1^* is derived from $\theta_1^* = \arg\min_\theta L(f_\theta(G_m^1))$, where $G_p \cap G_m^1 \neq \emptyset$; on the other hand, the calculation of $\hat{G}_p$ is based on θ_1^*. This interdependence further makes the optimization difficult to solve.

To this end, we propose to regard $\min_{\hat{G}_p} 1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$ as a clean-label poisoning attack, wherein the attacker adds elaborate poisoning examples to the training dataset to manipulate the behavior of the poisoned target model at the test time. In the context of this paper, $\mathcal{A}(f_{\theta_1^*}(\cdot))$ represents the poisoned model, and G_p is the poisoning graph data, whose goal of G_p is to make the poisoned model $\mathcal{A}(f_{\theta_1^*}(\cdot))$ classifies $\hat{G}_p$ as a specific target label 1 (member). According to the existing literature, the *feature collision* method [51] is an effective solution for this clean-label poisoning attack, whose attack paradigm can be instantiated considering the context of this paper as

$$\min_{G_p} D(f_{\theta_1^*}(G_p), f_{\theta_1^*}(\hat{G}_p)), \tag{10}$$

where $G_p = (A_p, X_p)$ is the radioactive graph, $\hat{G}_p = (\hat{A}_p, X_p)$ represents the querying version of G_p with isolated node sets (i.e., $\hat{A}_p = I_{|V_{G_p}|}$), and $D(\cdot, \cdot)$ represent the distance function. Note that the feature collision [51] has shown transferability for different classifiers (i.e., $\mathcal{A}$ in our case) [51], which underpins neglecting the impact of $\mathcal{A}$ in Equation (10).

Case-B: $\hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$. Similarly, we propose to consider $\min_{G_p} \hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$ as an evasion attack, where the original graph data of data owner is perturbed as G_p to make the target model $\hat{\mathcal{A}}(f_{\theta_0^*}(\cdot))$ predict 0 in $\hat{G}_p$. Note that $\hat{\mathcal{A}}(f_\theta(G)_{v_i}) = f_\theta(G)_{v_i}[y_i]$ as shown in Equation (7), thus $\min_{G_p} \hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$ can be transferred as

$$\max_{G_p} L(f_{\theta_0^*}(\hat{G}_p), Y_p) \tag{11}$$

where Y_p represents the ground-truth labels of nodes in $\hat{G}_p$, and $L(\cdot, \cdot)$ indicates the loss function.

(2) Solution Integration. To combine the above optimization solutions together, we propose a straightforward but effective approach called *feature-loss-max*, which can unify Equations (10) and (11) in the same view and then make solving the whole optimization (9) feasible. The intuition behind our *feature-loss-max* is that a machine learning model will preferentially fit training samples with larger prediction loss when all samples are treated equally. This intuition indicates that improving the prediction loss of f_{θ^*} in G_p contributes to improving the influence of G_p on θ^*, which helps improve the difference between θ_1^* and θ_0^* and then facilitates the distinguishing of $f_{\theta_1^*}$ and $f_{\theta_0^*}$. Next, we will introduce how to project Equations (10) and (11) into the view of feature (i.e., X_p in G_p) optimization.

Case-A: Optimization (10) suggests that the constructed G_p will make the target model $f_{\theta_1^*}$ focus on the node features but not on the graph structure, since $G_p = (A_p, X_p)$ and

$\hat{G}_p = (I_{|V_{G_p}|}, X_p)$. Therefore, we propose to mainly modify X_p when optimizing (10), i.e., $\min_{X_p} D(f_{\theta_1^*}(G_p), f_{\theta_1^*}(\hat{G}_p))$. We also note that: **(a)** According to the (D_1, D_2)-Lipschitz property [77] of the GNN models, we obtain the following.

$$D_1(f_{\theta_1^*}(G_p), f_{\theta_1^*}(\hat{G}_p)) \leq L_{f^c_{\theta_1^*}} D_2(f^e_{\theta_1^*}(G_p), f^e_{\theta_1^*}(\hat{G}_p)),$$

where $L_{f^c_{\theta_1^*}}$ indicates a constant scalar decided by $f^c_{\theta_1^*}$, D_1 and D_2 represents two distance functions, $f^c_{\theta_1^*}$ and $f^e_{\theta_1^*}$ indicate the classifier and encoder in the target GNN and $f_{\theta_1^*} = f^c_{\theta_1^*} \circ f^e_{\theta_1^*}$, symbol $\circ$ represents the function composition operation. Therefore, we conclude that reducing the difference between $f^e_{\theta_1^*}(G_p)$ and $f^e_{\theta_1^*}(\hat{G}_p)$ is equivalent to solving the optimization (10). **(b)** To avoid exposing the graph structure, we cannot deliver G_p to $f^e_{\theta_1^*}$. Thus, we propose to employ a pre-trained encoder $f^e_{\theta^p}$ to perform as $f^e_{\theta_1^*}$. In summary, our final optimization for the item $1 - \hat{\mathcal{A}}(f_{\theta_1^*}(\hat{G}_p))$ in Equation (9) is as follows.

$$\min_{X_p} \; ||f^e_{\theta^p}(G_p) - f^e_{\theta^p}(\hat{G}_p)||_2 \tag{12}$$

where $||\cdot||_2$ indicates the the ℓ_2-norm operation.

Case-B: Note that the nodes in $\hat{G}_p$ are isolated (i.e., $\hat{A}_p = I_{|V_{G_p}|}$), only the shared node features between $\hat{G}_p$ and G_p are optimized when solving Equation (11). Simultaneously, a surrogate model is needed when the data owner cannot ensure whether the target GNN f_{θ^*} is $f_{\theta_0^*}$. Thus, our optimization for the item $\hat{\mathcal{A}}(f_{\theta_0^*}(\hat{G}_p))$ in Equation (9) is

$$\max_{X_p} L(f_{\theta^p}(\hat{G}_p), Y_p), \tag{13}$$

where $f_{\theta^p} = f^c_{\theta^p} \circ f^e_{\theta^p}$ is a pre-trained surrogate model, which is consistent with prior works on graph unlearning [11]. Note that f_{θ^p} can be any publicly available GNN model that performs similar tasks or a handy GNN model trained on small-scale graph data.

Integration (A + B): Integrating the Equations (12) and (13) together, the radioactive graph in this paper is obtained by solving

$$\min_{X_p} \; ||f^e_{\theta^p}(G_p) - f^e_{\theta^p}(\hat{G}_p)||_2 - L(f_{\theta^p}(\hat{G}_p), Y_p). \tag{14}$$

Intuitively, the constructed $G_p = (A_p, X_p)$ will make the GNN trained in it (1) learn less about confidential A_p to protect the privatization of the data owner, and (2) produce a greater training loss in G_p, which increases its influence on the target GNNs and then facilitates the misuse detection.

ARTIFACT APPENDIX

This artifact is based on PyTorch and the deep graph library (DGL), and does not require GPU support. We implemented the proposed mitigation pipeline against data misuse in GNNs, including Proactive Misuse Detection and Training-graph-free Unlearning of our paper "GraphGuard: Detecting and Counteracting Training Data Misuse in Graph Neural Networks". The artifact can reproduce our major experimental results (AUC of the data misuse detection, visualization of the diverse output confidence scores distributions between benign/misused GNNs, accuracy and MIA attack success rate of GNNs after unlearning) reported in the main body of the paper. Our source code is available at https://github.com/GraphGuard/GraphGuard-Proactive/README.md, with a detailed guide at https://github.com/GraphGuard/GraphGuard-Proactive/AE/reproduce.md . We also uploaded the artifact to Zenodo, and the DOI is *https://doi.org/10.5281/zenodo.3549020*.

A. Description & Requirements

1) Security, privacy, and ethical concerns: None. All data misuse attacks against GNNs in our artifact are conducted on the simulation level, therefore do not lead to any damage in the real world.

2) How to access: Our artifact source code is hosted in a GitHub repository, available through https://github.com/GraphGuard/GraphGuard-Proactive/README.md.

3) Hardware dependencies: This artifact requires a Linux server and does not require GPU support.

4) Software dependencies: This artifact relies on multiple existing Python packages, including Python, PyTorch, DGL, and so on (details in `Init environment` section in `README.md`). Our guide includes all the details to set up the required software dependencies.

5) Benchmarks: Our experiments with this artifact are reported on 4 benchmark datasets: Cora, Citeseer, Pubmed (three citation networks commonly used for evaluating node classification tasks in GNNs), Flickr (an image dataset with profiles, where a node represents an image, and an edge represents two images have common information). All of them are integrated into DGL and automatically downloaded and setup in our artifact.

B. Artifact Installation & Configuration

Our documentation contains a detailed guide for installing our artifact and required environments. Briefly, the installation procedure is as follows.

1 Get the artifact from https://github.com/GraphGuard/GraphGuard-Proactive.

2 Install Conda following the instructions in https://conda.io/projects/conda/en/latest/user-guide/install/index.html to install required environments.

3 Install Pytorach and other necessary Python packages using instructions in `Init environment` section in `README.md`.

C. Major Claims

- (C1): Proactive Misuse Detection in our proposed design *GraphGuard* can strengthen the ability to discern benign and misused model performance. This is proven by the experiment (E2) whose results are illustrated in Table IV. It also achieves better detection performance (higher AUC) compared to existing studies. This is proven by the experiment (E1) whose results are illustrated in Figure 4.
- (C2): Training-graph-free Unlearning in *GraphGuard* can mitigate data misuse in GNNs and decrease the MIA successful rate of unlearned nodes before/after unlearning. This is proven by the experiment (E3) whose results are illustrated in Table VII. It will also not lead to significant deduction of the model accuracy. These are proven by the experiment (E4) whose results are illustrated in Table VI.

D. Evaluation

1) Experiment (E1): [Effectiveness of Proactive Misuse Detection]: Experiment (E1) evaluates the effectiveness of our proposed Proactive Misused Detection and compares it to the baseline membership inference attacks (MIAs) across 4 different models and datasets. It calculates the AUC of both our detection and the baseline detection via MIAs, thus proving our first claim (C1). Experiment (E1) corresponds to our reported results in Table IV.

[How to] All the preparation steps have been stated in Artifact Appendix B. We provide all the necessary commands to reproduce our results of (E1) in `AE/reproduce.md#Results-for-E1`.

[Results] The AUC of our proposed detection is almost 1, therefore proving that our design is highly effective in detecting data misuse. And the AUC of the baseline MIA is less than our proposed design.

2) Experiment (E2): [Output Distribution Comparisons]: Experiment (E2) evaluates the ability of our proposed Proactive Misused Detection about how it can discern the output distribution of benign and misused models. It compares the output distribution across 4 different models and datasets, therefore proving our first claim (C1). Experiment (E2) corresponds to our reported results in Figure 4.

[How to] All the preparation steps have been stated in Artifact Appendix B. We provide all the necessary commands to reproduce our results of (E2) in `AE/reproduce.md#Results-for-E2`.

[Results] The difference between the output distribution of benign/misused models with our design is significant, while there is an overlap between the output distribution of benign/misused models without our design.

3) Experiment (E3): [Effectiveness of Training-graph-free Unlearning] Experiment (E3) evaluates the effectiveness of our proposed Training-Graph-Free Unlearning. It calculates the MIAs of the data-misused model with/without using unlearning, which proves our second claim (C2). The results correspond to the results reported in Table VII.

[How to] All the preparation steps have been stated in Artifact Appendix B. We provide all the necessary commands to reproduce our results of (E3) in `AE/reproduce.md#Results-for-E3`.

[Results] The attack success rate of the MIAs is significantly reduced after using our unlearning method.

4) Experiment (E4): [Utility of Training-graph-free Unlearning] Experiment (E4) evaluates the utility of our proposed Training-Graph-Free Unlearning. It also compares the accuracy of the unlearned model and the retrained model as a baseline. The results correspond to the results reported in Table VI.

[How to] All the preparation steps have been stated in Artifact Appendix B. We provide all the commands necessary to reproduce our results of (E4) in `AE/reproduce.md#Results-for-E4`.

[Results] The accuracy of our unlearned model is only slightly lower than the retraining baseline (less than 5%).