Devil in Disguise: Breaching Graph Neural Networks Privacy through Infiltration

Lingshuo Meng
Zhejiang University
emeng@zju.edu.cn

Yijie Bai
Zhejiang University
baiyj@zju.edu.cn

Yanjiao Chen*
Zhejiang University
chenyanjiao@zju.edu.cn

Yutong Hu
Zhejiang University
hytong@zju.edu.cn

Wenyuan Xu
Zhejiang University
wyxu@zju.edu.cn

Haiqin Weng
Ant Group
haiqin.wenghaiqin@antgroup.com

ABSTRACT

Graph neural networks (GNNs) have been developed to mine useful information from graph data of various applications, e.g., healthcare, fraud detection, and social recommendation. However, GNNs open up new attack surfaces for privacy attacks on graph data. In this paper, we propose Infiltrator, a privacy attack that is able to pry node-level private information based on black-box access to GNNs. Different from existing works that require prior information of the victim node, we explore the possibility of conducting the attack without any information of the victim node. Our idea is to infiltrate the graph with attacker-created nodes to befriend the victim node. More specifically, we design infiltration schemes that enable the adversary to infer the label, neighboring links, and sensitive attributes of a victim node. We evaluate Infiltrator with extensive experiments on three representative GNN models and six real-world datasets. The results demonstrate that Infiltrator can achieve an attack performance of more than 98% in all three attacks, outperforming baseline approaches. We further evaluate the defense resistance of Infiltrator against the graph homophily defender and the differentially private model.

CCS CONCEPTS

• **Security and privacy → Software and application security**;
• **Computing methodologies → Neural networks**.

KEYWORDS

Graph neural network; Machine learning privacy; Inference attack

ACM Reference Format:
Lingshuo Meng, Yijie Bai, Yanjiao Chen, Yutong Hu, Wenyuan Xu, and Haiqin Weng. 2023. Devil in Disguise: Breaching Graph Neural Networks Privacy through Infiltration. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623173

*Corresponding author.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3623173

1 INTRODUCTION

Graph becomes a prevalent data format for many real-world applications. For instance, a social network is often abstracted as a graph, where nodes represent users and links indicate social relationships. Graph data shows rich expressiveness and may carry private information of users. For instance, user profiles and follower relationships in social networks are intriguing for system developers and attackers [5, 48, 54].

To fully exploit the information of graph data, graph neural networks (GNNs) have been proposed [12, 22, 42, 50, 59]. GNNs effectively integrate the high-dimensional node features and high-order link structures of the graph data, resulting in advanced performance compared with traditional methods. Specifically, a GNN adopts a message passing mechanism, by which the nodes aggregate messages from neighboring nodes through graph links to update their representations. Due to their superiority in graph data modeling, GNNs enable a wide range of prevalent applications, such as health care [61, 63], fraud detection [24], and social recommendation [10, 15]. However, the emergence of GNNs also intensifies the privacy threats to graph data.

Recent works have studied privacy attacks against GNN models [13, 14, 45, 47, 58]. However, these works mainly focus on *graph-level* private information, e.g., the total number of nodes/links contained in the graph [58]. Compared with graph-level information, node-level information of a graph is more abundant and privacy-intruding, e.g., sensitive attributes of a certain node. A few works have investigated link inference attacks [13, 47], but they require certain information of the victim node (detailed discussion in Section 2.2). Overall, there is a lack of systematic privacy attacks to infer fine-grained node-level information based on only black-box access to the GNN model.

In this paper, we propose a holistic attack framework, dubbed Infiltrator, which reveals sensitive information of a victim node with black-box access to GNNs. We consider a practical setting where the adversary has no access to any information of the victim node, which is much stricter than existing attacks [13, 14, 52]. To fulfill this goal, a set of nodes and links is created and infiltrated into the graph. Through careful designs, the target GNN will unconsciously provide the posterior, which contains private information of the victim node. In particular, Infiltrator consists of three modules. The attack scenario of Infiltrator is demonstrated in Figure 1.

- **Label-Infiltrator.** Label-Infiltrator aims to infer the label of the victim node through minimal infiltration. With the

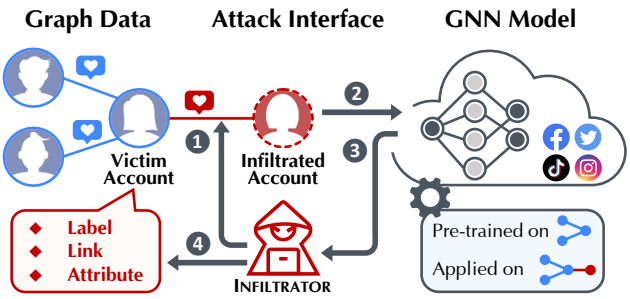

Figure 1: Attack Scenario of INFILTRATOR. The adversary (1) infiltrates the graph data, (2) queries the target GNN model, (3) receives the output posterior, and (4) discloses the confidential label, links, and attributes of the victim node.

minimal prior knowledge of the victim node, the adversary injects only a single node and a single link. Through an affiliated gradient distance estimation, we further enhance the inference performance.

- **Link-INFILTRATOR.** Link-INFILTRATOR performs a link inference that infers all neighboring nodes of a victim node. Through dynamic infiltration among candidate nodes, we can derive the possible links/neighbors without a pre-calibrated threshold.
- **Attribute-INFILTRATOR.** Attribute-INFILTRATOR is the most challenging attack that endeavors to infer the sensitive attributes (discrete and continuous) from the feature set of the victim node through symmetric infiltration. We formulate the problem as an optimization task and solve it in a black-box setting.

We evaluate the effectiveness of INFILTRATOR on three representative GNN models [12, 22, 42] and six real-world graph datasets. It is confirmed that INFILTRATOR outperforms all baselines, with up to 98% attack accuracy on citation graphs and large-scale social networks. We further demonstrate that INFILTRATOR remains to be effective when we transfer the attack model to unseen data. Furthermore, we investigate the defense resistance of INFILTRATOR against the graph homophily defender and the GNN protected with differential privacy.

2 PRELIMINARIES

2.1 Graph Neural Networks

Graph neural networks are a family of neural networks for processing data that can be represented as a graph. We define an undirected graph[1] as $G = (V, E)$, where V denotes the set of nodes $v \in V$, and $e_{vu} \in E$ denotes the set of edges, where $e_{vu} = (v, u)$ denotes the existence of a link between v and u. We further define the neighborhood of a node v as $\mathcal{N}_v = \{u \in V | (v, u) \in E\}$. Given an input graph G, a GNN models its topological structures $\mathbf{A}$ and descriptive features $\mathbf{X}$. $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$ is the adjacency matrix of G, where $\mathbf{A}_{vu} = 1$ if $e_{vu} \in E$; otherwise $\mathbf{A}_{vu} = 0$. $\mathbf{X} \in \mathbb{R}^{|V| \times D}$ consists of attribute vectors of nodes, where $\mathbf{x}_v \in \mathbf{X}$ is a D-dimensional attribute vector of node v.

[1]By symmetrizing the edges, GNNs tend to preprocess directed graphs into undirected graphs for improved performance [42, 56].

GNNs can be categorized into recurrent graph neural networks (RecGNN), convolutional graph neural networks (ConvGNN), graph autoencoders, and spatial-temporal graph neural networks [48]. RecGNNs are early works of GNNs, inspiring later works on ConvGNNs. Graph autoencoders learn the node/graph representations through unsupervised learning. Spatial-temporal GNNs deal with spatial-temporal graph data. In this paper, we focus on ConvGNNs that apply convolutions to graph data, which have gained wide popularity in recent years as convolutional operations are easier to be integrated with other neural networks.

ConvGNNs usually process the graph input in a *message passing* manner, by which every node receives and aggregates messages from neighboring nodes to update its representation through neural networks. During the l-th message passing (i.e., forwarding through the l-th GNN layer), node v updates the previous embedding $\mathbf{h}_v^{l-1}$ with the message $\mathbf{m}_v^{l-1}$ that aggregates embeddings from the neighborhood $\mathcal{N}_v$,

$$\mathbf{h}_v^l = \text{Update}^l\left(\mathbf{h}_v^{l-1}, \mathbf{m}_v^{l-1}\right), \tag{1}$$

$$\mathbf{m}_v^{l-1} = \text{Aggregate}^l\left(\left\{\mathbf{h}_u^{l-1}, \forall u \in \mathcal{N}_v\right\}\right) \tag{2}$$

where $\text{Aggregate}(\cdot)$ and $\text{Update}(\cdot)$ are differentiable operations with (potentially) learnable parameters. The initial embedding $\mathbf{h}_v^0$ is set as $\mathbf{h}_v^0 = \mathbf{x}_v$. After iterating L times message passing, the embedding of a node captures the information from all nodes up to L hops away.

After node embeddings are obtained, various downstream tasks can be performed, such as node classification [22, 27] and graph classification [53, 55]. Node classification tasks aim to predict labels of individual nodes in the graph, e.g., tagging the community of a user in a social network. Graph classification tasks aim to predict the label of the entire graph, e.g., determining the cellular function of proteins. To perform graph classification, the node embeddings will be pooled together with a pooling/readout layer to form the graph representation. In this paper, we focus on node classification tasks and discuss the possible extension to graph classification tasks in Appendix F (in our full version).

The training process of GNN aims to minimize the loss function,

$$\arg\min_{\theta} \mathcal{L}\left(f\left(\mathbf{X}, \mathbf{A}; \theta\right), \mathbf{Y}\right), \tag{3}$$

where $\mathcal{L}$ quantifies the classification loss (e.g., cross-entropy loss), f is a GNN model parameterized by θ, $\mathbf{Y}$ is the corresponding labels of nodes.

GNNs may be trained in ways of transductive learning or inductive learning.

- **Transductive learning.** In transductive learning, GNNs are trained in a semi-supervised manner. More specifically, both labeled and unlabeled nodes of a fixed graph are used to train a GNN, which is then used to predict the label of unlabeled nodes in this graph. If new nodes are introduced, the GNN needs to be retrained, thus transductive learning has poor generalizability.
- **Inductive learning.** In inductive learning, GNNs are trained on labeled nodes and can be used to predict the labels of nodes unseen during the training process.

In this paper, we focus on inductive learning which is more practical for real-world graph data analytics [12, 47] due to its generalizability.

2.2 Privacy Threats Against GNNs

A graph is composed of nodes and links whose information is deemed crucial and private. For example, social networks are often represented as a graph, with nodes denoting user accounts and links denoting social relations. Although such information often requires access control (e.g., Twitter[2] or Facebook[3]), adversaries are motivated to pry the private information by various techniques.

An adversary may target *graph-level* or *node/link-level* private information of a graph. Graph-level information refers to global information of the graph, e.g., the total number of nodes/links in the graph or the existence of certain nodes/links in the graph. Node/link-level refers to information of individual nodes or links, e.g., node features and the existence of a link between a node pair. In this paper, we focus on more fine-grained node-level information[4]. More specifically, we consider three potential privacy threats in GNNs, i.e., *node label inference attacks*, *node-centric link inference attacks*, and *node attribute inference attacks*. As far as we know, there are no existing works on node label and node attribute inference attacks against GNNs, and only a few existing works on link inference attacks against GNNs [13, 47]. Existing link inference attacks against GNNs can be categorized into two classes.

Similarity-based attacks. Similarity-based attacks [13] are based on the intuition that two nodes are more likely to be linked if they are more *similar*. Link Stealing Attack (LSA) is a similarity-based attack that quantifies the similarity of two nodes based on various distance functions of their GNN posteriors, among which *correlation distance* is shown to be the best,

$$dis(\mathbf{u}, \mathbf{v}) = 1 - \frac{(\mathbf{u} - \overline{\mathbf{u}}) \cdot (\mathbf{v} - \overline{\mathbf{v}})}{\|\mathbf{u} - \overline{\mathbf{u}}\|_2 \cdot \|\mathbf{v} - \overline{\mathbf{v}}\|_2}, \tag{4}$$

where $\mathbf{u}, \mathbf{v}$ are GNN posteriors for nodes u and v. Given the distance values between all node pairs, those with the lowest distance values can be estimated as being linked. An alternative way to determine linked nodes is to cluster node pairs into two groups based on distance values and identify the group with the lower average distance as linked node pairs.

Influence-based attacks. Influence-based attacks [47] are based on the assumption that two nodes are more likely to be linked if changing the features of one node highly influences the prediction results of the other node, which is because of the message passing mechanism in GNN. LinkTeller (LTA) is an influence-based attack that builds the influence matrix of node v as

$$\mathbf{I}_v = \lim_{\delta \to 0} \frac{f(\mathbf{X} + \Delta) - f(\mathbf{X})}{\delta}, \tag{5}$$

where $\Delta = \left[0, 0, \ldots, \delta \cdot \mathbf{x}_v^\top, \ldots, 0\right]^\top$. The u-th row $\mathbf{I}_{vu} = \mathbf{I}_v[u, :]$ represents the influence of node v on node u regarding all output classes of GNN. The final influence value of node v on node u is calculated by the L_2 norm of $\mathbf{I}_{vu}$, i.e., $\|\mathbf{I}_{vu}\|_2$. The node pairs with the highest influence values are considered to be linked.

2.3 Threat Model

We portray the threat model in terms of the knowledge, capability, and goal of the adversary. Without loss of generality, we showcase a social network that is represented as a graph and a GNN that performs the node classification task based on the graph.

Attacker's knowledge. We consider a black-box scenario where the adversary has no knowledge of the target GNN, including its parameters, hyperparameters, and training data. Moreover, the adversary does not know any information of the victim node[5], including its features and posterior output of the GNN.

Attacker's capability. We define the attacker's capability from the following two aspects.

(1) *Access posteriors.* The adversary is able to query the GNN and receive corresponding prediction results. This is supported by many commercial APIs customized for graph learning, such as Neptune ML[6] developed by Amazon. We assume prediction results in the form of the posterior distribution (i.e., posteriors). Obtaining posteriors has been widely adopted by recent privacy attacks against GNNs [13, 35, 45].

(2) *Perform Infiltration.* The adversary can also perform infiltration, e.g., adding nodes into the graph by creating user accounts in the social network. We restrict the number of attacker-added nodes to no more than 1, 2, and 3 in Label-, Link-, and Attribute-Infiltrator. Performing graph modifications during the inference time has been widely adopted in existing GNN evasion attacks, and the limited modification by Infiltrator is minute compared to existing attacks [44, 64, 65].

Attacker's goal. We consider that the goal of the attacker is to infer the node label, node neighbors, and node attributes.

- **Label inference attack (Label-Infiltrator).** The goal is to infer the label y_v of the victim node v. The label y_v may capture sensitive information of user v, e.g., the recommendation for users that may disclose their interests. Label inference attacks may also serve as the basis to facilitate further attacks.
- **Link inference attack (Link-Infiltrator).** The goal is to infer the neighbor set $\mathcal{N}_v$ of the victim node v, which consists of nodes with a link to the victim node. The link information may reveal the sensitive social relations of the victim user. The link inference attacks can base on known label y_v of the victim node, e.g., by Label-Infiltrator.
- **Attribute inference attack (Attribute-Infiltrator).** The goal is to infer sensitive attribute of the victim node. The feature vector $\mathbf{x}_v$ of the victim node can be divided into sensitive attributes $\mathcal{S}_v$ and non-sensitive attributes $\overline{\mathcal{S}}_v = \mathbf{x}_v \setminus \mathcal{S}_v$. Sensitive attributes of the victim nodes may be race, gender, or occupation of the victim user. Note that Attribute-Infiltrator differs from property attacks against GNNs [45, 58] that target attributes not contained in the feature vector. The attribute inference attacks are based on known label and neighbors of the victim node, e.g.,

[2] https://developer.twitter.com/docs/twitter-api
[3] https://developers.facebook.com/docs/graph-api
[4] Link-Infiltrator can also cover link-level information. We refer interested readers to [9, 45, 58] for privacy attacks targeting graph-level information in GNN.

[5] For the most challenging Attribute-Infiltrator that relies on the neighbors of the victim node, we investigate its performance under imprecise neighbor information in Section 4.3.3.
[6] https://aws.amazon.com/neptune/machine-learning

by Label-Infiltrator and Link-Infiltrator. Following existing works [17, 26], we assume that non-sensitive attributes $\overline{S}_v$ are also known to the adversary.

3 INFILTRATOR: DETAILED DESIGN

Given black-box access to the GNN model f, Infiltrator aims to reveal label-, link- and attribute-wise information of a victim node v. An overview of Infiltrator is shown in Figure 2.

3.1 Label-Infiltrator

Attack overview. Label-Infiltrator is the foundation of Infiltrator where the adversary knows nothing about the victim node. Label-Infiltrator may serve as the springboard for the following Link-Infiltrator and Attribute-Infiltrator. Figure 2(a) illustrates the overview of Label-Infiltrator.

Infiltration scheme. Label-Infiltrator performs minimal infiltration. The adversarial infiltration set consists of a single infiltrator node $V_{adv} = \{a\}$ and the link connects with the victim $E_{adv} = \{(a, v)\}$. The adversary may establish (a, v) by following the victim in the social network. The only information available for the adversary is the GNN's posterior of the infiltrator node p_a. Due to the message passing mechanism of GNN, the posteriors of the infiltrator node a and the victim node v are highly correlated. According to Equations (1) and (2), the output of node a depends on its own feature $\mathbf{x}_a$ and the information from its neighbors. For the infiltrator node controlled by the adversary, its only neighbor is the victim node v. Therefore, the output p_a contains essential information about the victim node that helps with inferring the label of the victim node. To reduce the influence of $\mathbf{x}_a$ on p_a, we set $\mathbf{x}_a$ as $\mathbf{0}$ rather than a random vector. A nil $\mathbf{x}_a$ also minimizes the impact of infiltration on the victim node. We investigate the impact of non-nil infiltrator node attributes on Infiltrator in Section 4.3.4.

Label inference. Given the posterior p_a, a naive way is to use the label deduced from p_a as the inferred label of the victim node, i.e., $y_v = y_a = \arg\max p_a$. Our experiments show that this simple method is effective in most cases, except when the top-1 and top-2 confidence scores in p_a are close to each other, which indicates the underfitting prediction of GNNs on the infiltrator node. To deal with this problem, we propose a gradient-based label inference algorithm. The gradient of the loss implies the gap toward the optimal training results. Under similar confidence scores, the class with a larger gradient can achieve a relatively high confidence score without much training effort and is likely to reflect the true label. In the black-box setting, we estimate the gradient of the loss function $\mathcal{L}$ regarding each class c using zeroth order estimation [4],

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_a} \doteq \lim_{\delta \to 0} \frac{\mathcal{L}(p'_a) - \mathcal{L}(p_a)}{\delta}, \tag{6}$$

where $p'_a = f(\mathbf{x}_a + \delta \cdot \mathbf{e})$, $\mathbf{e}$ is a unit feature vector. δ is a small perturbation scale, which is set to 1e-4 for continuous attributes and to 1 for discrete attributes. We summarize the label inference process in Algorithm 1.

Algorithm 1: Label-Infiltrator Algorithm

Input: Inductive GNN model f, the posterior of the infiltrator node p_a.
Output: The label of the victim node y_v.

```
/* Confidence Score Inference                    */
```
$(p_1, y_1) = \arg\max_c p_a$.
// p_1 is the top-1 confidence score and y_1 is the corresponding label.
$(p_2, y_2) = \arg\max_c p_a \setminus p_1$.
// p_2 is the top-2 confidence score and y_2 is the corresponding label.
if $p_1 > \alpha \cdot p_2$ **then**
  // α controls the comparison bound.
  return $y_v = y_1$.
else
  Class set $C = \{c | \alpha \cdot p_c \geq p_1\}$.
  // consider all classes with a confidence score close to p_1.
end
```
/* Gradient Estimation Inference                 */
```
$p'_a = f(\mathbf{x}_a + \delta \cdot \mathbf{e})$.
foreach class c in C **do**
  Calculate ∇ as in Equation (6).
end
return $y_v = \arg\max_c \nabla$.

3.2 Link-Infiltrator

Attack overview. Link-Infiltrator infers potential neighbors linked to the victim node v. We use U to denote a candidate set of nodes of which the link to the victim node v we are interested in. Note that Link-Infiltrator can be easily extended to infer the existence of links between any pair of nodes in the graph. Figure 2(b) illustrates the overview of Link-Infiltrator.

Infiltration scheme. Link-Infiltrator performs infiltration with two nodes $V_{adv} = \{a, b\}$ and two links $E_{adv} = \{(a, v), (b, u)\}$, where $u \in U$ is a candidate node. The intuition of Link-Infiltrator is that if victim node v and candidate node u are connected by a link, the information of u will transmit to v through the message passing mechanism of GNN and impact the posterior p_v, vice versa. As the adversary does not have access to the posterior of v, we approximate p_v with p_a with the same rationale as in Label-Infiltrator.

Link inference. Link-Infiltrator is inspired by the influence-base attack [47], which captures the influence of the victim node v on the candidate node u by perturbing the attributes of v and monitoring the changes in p_u. Nonetheless, this may not be realistic as (1) the adversary may not be able to alter the attributes of the victim node, and (2) the perturbations to the attributes need to be small, which may lead to illegal values, e.g., when discrete attributes are required. To address this issue, we propose to reversely evaluate the influence of the candidate node u on the victim node by monitoring the posterior change of p_v before and after infiltration,

$$\mathbf{RI}_u = p_v(\mathbf{X}^{u+}, \mathbf{A}^{u+}) - p_v(\mathbf{X}, \mathbf{A}), \tag{7}$$

where $(\mathbf{X}^{u+}, \mathbf{A}^{u+})$ represents the graph after the infiltration on the candidate node side rather than on the victim side, and $p_v(\cdot)$ represents the posteriors of the victim v predicted by GNNs. Note that for any linked pair (u, v), the infiltration should yield large influence

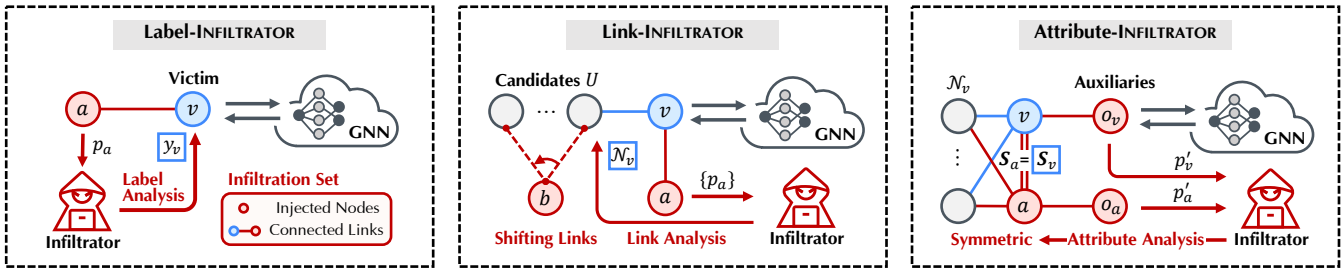

Figure 2: Overview of Infiltrator. Given black-box access to the target GNN model and a victim node v, (a) Label-Infiltrator aims to infer the victim's label y_v, (b) Link-Infiltrator aims to infer the victim's links/neighbors $\mathcal{N}_v$, (c) Attribute-Infiltrator aims to infer the victim's sensitive attribute(s) $\mathcal{S}_v$.

Algorithm 2: Link-Infiltrator Algorithm

Input: Candidate set U, graph data $(\mathbf{X}, \mathbf{A})$, infiltrator nodes $\{a, b\}$, GNN posteriors p_a for a.
Output: Estimated neighbors of the victim node $\hat{\mathcal{N}}_v$.

1 $\hat{\mathcal{N}}_v \leftarrow \emptyset$.
/* 1. Anchor Score: Estimate $p_v(\mathbf{X}, \mathbf{A})$ */
2 Perform the infiltration of (a, v).
3 $p_{a,0} = p_a$.
/* 2. Influenced Score: Estimate $p_v(\mathbf{X}^{u+}, \mathbf{A}^{u+})$ */
4 **foreach** node u in U **do**
5 Shift the infiltration of (b, u).
6 $p_{a,u} = p_a$.
7 **if** $||p_{a,u} - p_{a,0}||_2 > \theta$ **then**
// θ determines the likelihood of $u \in \mathcal{N}_v$.
8 $\hat{\mathcal{N}}_v \leftarrow \hat{\mathcal{N}}_v \cup \{u\}$.
9 **end**
10 **end**
11 **return** $\hat{\mathcal{N}}_v$.

of $||\mathbf{RI}_u||_2$. As the adversary only has access to the nodes in V_{adv}, we estimate the influence by performing two-step infiltration, as summarized in Algorithm 2.

To estimate the original p_v, we first create infiltrator node a and link (a, v) to obtain an anchor posterior $p_{a,0}$ (Line 2-3). Then, we create infiltrator node b and link (b, u) to obtain the influenced posterior $p_{a,u}$ for each u (Line 4-6). For $u \in \mathcal{N}_v$, the infiltration of (b, u) will affect the posterior $p_{a,u}$ greater than for $u \notin \mathcal{N}_v$. If the estimated influence of $||\mathbf{RI}_u||_2$, i.e., $||p_{a,u} - p_{a,0}||_2 > \theta$, we determine that candidate node u is likely to be a neighbor of the victim node v. Note that the addition of a and b may also affect the information interaction between u and v, but we set $\mathbf{x}_a = \mathbf{x}_b = \mathbf{0}$ to mitigate the unexpected influence. During implementation, we set θ as 1e-7, which achieves high precision. Moreover, we shift the dynamic link (b, u) asynchronously also to mitigate the influence on each u.

3.3 Attribute-Infiltrator

Attack overview. Attribute-Infiltrator aims to learn the sensitive attributes $\mathcal{S}_v$ from the feature vector of the victim $\mathbf{x}_v$. Attribute-Infiltrator is the most challenging since the sensitive attributes are heterogeneous in value type and range. Attribute-Infiltrator can rely on the results of Label-Infiltrator and Link-Infiltrator. Figure 2(c) illustrates the overview for Attribute-Infiltrator.

Infiltration scheme. Attribute-Infiltrator performs infiltration with three additional nodes $V_{adv} = \{a, o_a, o_v\}$ and the link set $E_{adv} = \{(a, u) | u \in \mathcal{N}_v\} \cup \{(o_a, a), (o_v, v)\}$. As shown in Figure 2(c), we form a symmetric structure where infiltrator node a and victim node v are connected to the same set of neighbors. Node o_a and node o_v serve as auxiliary nodes for extracting information from a and v, respectively. The rationale is that if infiltrator node a has the same sensitive attributes as victim node v, their GNN outputs p_a and p_v should be similar due to the permutation[7] equivariance of GNNs.

Definition 3.1 (Permutation Invariance [21]). For any permutation ρ, a function f is invariant to graph input G if:

$$f(\rho \star G) = f(G).$$

Definition 3.2 (Permutation Equivariance [21]). For any permutation ρ, a function f is equivariant to graph input G if:

$$f(\rho \star G) = \rho \star f(G).$$

GNN models are considered to be equivariant to permutation in node classification tasks. If the topological structures and attributes of any two nodes remain consistent, the GNN will return equivalent results regardless of the permutation. The infiltrator node a is created as a symmetric counterpart for the victim node v. We adjust the sensitive attributes $\mathcal{S}_a$ of node a to try to make p_a equate p_v. Since p_v is unknown to the adversary, we construct o_v, whose output p'_v approximates p_v. Accordingly, we establish o_a (whose output is p'_a) to maintain the symmetric structure between a and v.

Attribute inference. We formalize the attribute inference attack as an optimization problem,

$$\mathcal{S}_v^* = \arg\min_{\mathcal{S}_a} \ell_2(p'_a, p'_v), \tag{8}$$

where $\ell_2(\cdot)$ calculates the mean square error of the two posteriors. Recall that the adversary has only black-box access to the GNN model, which impedes feature-space optimization without model weights. To tackle this problem, we perform zeroth order optimization [4] to estimate the gradient of ℓ_2 as,

$$\frac{\partial \ell_2}{\partial \mathcal{S}_a} \doteq \lim_{\delta \to 0} \frac{\ell_2(\mathcal{S}_a + \delta \cdot \mathbf{e}_i) - \ell_2(\mathcal{S}_a - \delta \cdot \mathbf{e}_i)}{2 \cdot \delta}. \tag{9}$$

[7] Permutation ρ reorders the nodes of the graph G. We refer interested readers to [21] for the standard definition.

Algorithm 3: Attribute-Infiltrator Algorithm

Input: Sensitive attributes $\mathcal{S}$, initial feature $\mathbf{x}_a$ with non-sensitive attributes $\overline{\mathcal{S}}_v$, learning step η.
Output: Sensitive attributes of the victim node $\mathcal{S}_v$.

while not convergence **do**
  Sample dimension(s) i from $\mathcal{S}$.
  /* Gradient Estimation */
  Calculate ∇ as in Equation (9).
  if discrete features **then**
    $\nabla \leftarrow \text{Sign}(\nabla)$
    // consider the sign information to indicate the direction when optimizing discrete data.
  end
  /* Gradient Descent */
  $\mathcal{S}_a[i] \leftarrow \mathcal{S}_a[i] - \eta \cdot \nabla$
  // update the sampled i-dimensional sensitive attributes of $\mathcal{S}_a$.
end
return $\mathcal{S}_v = \mathcal{S}_a$.

We deem the loss ℓ_2 as a function of $\mathcal{S}_a$. Same as Equation (6), δ is the small perturbation scale. $\mathbf{e}_i$ is a hot vector with the i-th element as 1, where i is sampled from sensitive attributes $\mathcal{S}$. We summarize the optimization process in Algorithm 3. For discrete features, we set the learning step η to 1, which ensures that the discrete attributes take legal values stipulated by the GNN model. For continuous features, we set η as 1e-2. We further discuss other black-box optimization methods in Appendix E.2 and show that Attribute-Infiltrator outperforms these methods in query efficiency.

4 EVALUATION

In this section, we comprehensively assess the performance of Infiltrator through the following three research questions:

- **RQ1** - How effective is Infiltrator on representative GNNs and real-world datasets?
- **RQ2** - How robust is Infiltrator under different conditions (e.g., overfitting/deeper GNNs, evolving/lacking information)?
- **RQ3** - How well is the performance of Infiltrator when transferred with inductive GNNs?

4.1 Experiment Setup

Datasets. We conduct experiments on six real-world datasets, including three citation networks, i.e., Cora [51], Citeseer [51], Coauthor CS [33], as well as three social networks, i.e., Facebook [41], Twitch-ES [28], LastFM Asia [29]. We summarize the dataset statistics in Table 1. More details of the datasets are in Appendix B. We sample 75% of the datasets as the training dataset and use the entire graph for GNN inference.

GNN models. We consider three de facto standard GNN models, i.e., Graph Convolutional Networks (GCN) [22], GraphSAGE (SAGE) [12], and Graph Attention Networks (GAT) [42]. Concretely, we mainly consider the 2-layer GCN, 3-layer SAGE, and 3-layer GAT for the diversity of model architecture. We set 64 neurons in each hidden layer, with ReLU as the activation function. We adopt softmax in the output layer and use the cross-entropy loss for node classification. For each GNN model, we train the model for 200 epochs with suitable learning rates. Experiment results show that our GNNs achieve comparable performance on the training and testing datasets. More details of the GNN models are in Appendix A. We implement the GNN models in PyTorch using PyTorch-Geometric (PyG)[8]. Experiments are all carried out on an Ubuntu 20.04 system with Intel CPU of 40 cores, 128GB RAM, and 4 NVIDIA GeForce RTX 3090 Ti GPUs.

Table 1: Statistics of datasets. Attr.: Attributes. d: discrete attributes. c: continuous attributes.

Type	Dataset	# Nodes	# Edges	# Attr.	# Classes
Citation Network	Cora	2,708	10,556	1,433 (d)	7
	Citeseer	3,327	9,104	3,703 (d)	6
	Coauthor	18,333	163,788	6,805 (d)	15
Social Network	Facebook	17,440	1,116,810	288 (d)	5
	Twitch	4,648	123,412	128 (c)	2
	LastFM	7,624	55,612	128 (c)	18

Table 2: Attack performance of label inference attacks.

	GCN			SAGE			GAT		
	HG	MI	**Ours**	HG	MI	**Ours**	HG	MI	**Ours**
Cora	87.26%	93.36%	**98.06%**	86.71%	88.23%	**94.68%**	87.08%	85.80%	**97.15%**
Citeseer	75.56%	94.61%	**96.89%**	73.19%	74.79%	**81.24%**	75.51%	87.94%	**92.66%**
Coauthor	94.17%	98.43%	**99.30%**	91.97%	95.88%	**97.92%**	93.84%	94.51%	**97.08%**
Facebook	**96.74%**	60.64%	94.72%	94.46%	79.52%	**96.78%**	**96.08%**	84.11%	94.68%
Twitch	74.42%	81.36%	**93.78%**	68.85%	67.82%	**96.63%**	69.18%	77.71%	**95.82%**
LastFM	92.14%	90.77%	**96.99%**	91.12%	82.66%	**91.44%**	89.22%	81.32%	**91.88%**

4.2 RQ1: Effectiveness of Infiltrator

We evaluate the effectiveness of Infiltrator in comparison with potential baselines. Further, we provide the model accuracy under attacks to ensure that GNN utility can be well preserved.

4.2.1 Label-Infiltrator vs. Baselines. We compare the attack with two baselines.

- **Maximum Inference (MI).** MI is a naive version of Label-Infiltrator, where the adversary directly derives the label of the victim node as the one with the maximal confidence score, i.e., $y_v = \arg\max p_a$.
- **Homophily Guessing (HG).** Nodes are likely to have the same label as their neighbors, which is known as the homophily of graph networks [6, 25, 62]. We construct a baseline label inference attack based on homophily. Specifically, the adversary collects the labels of the neighborhood nodes $\mathcal{N}_v$. The adversary then guesses the label of the victim node through a majority vote. Note that Infiltrator has a stronger threat model than HG. The adversary of Infiltrator has no access to any label information, while the adversary of HG is assumed to have the label information of the neighbors of the victim node.

[8] https://www.pyg.org

Table 3: Attack performance of Link-Infiltrator. pre.: precision (%). rec.: recall (%).

Dataset		GCN		SAGE		GAT	
		pre.	rec.	pre.	rec.	pre.	rec.
Cora	LSA	0.66%	42.33%	0.66%	41.84%	0.66%	42.06%
	LTA	57.88%	64.74%	73.44%	81.93%	46.07%	51.64%
	Ours	**98.39%**	**99.99%**	**98.31%**	**98.86%**	**91.02%**	**91.46%**
Citeseer	LSA	0.60%	53.04%	0.50%	40.70%	0.61%	53.20%
	LTA	74.03%	82.43%	82.28%	91.42%	68.73%	76.84%
	Ours	**98.27%**	**99.84%**	**99.39%**	**99.75%**	**91.88%**	**93.35%**
Coauthor	LSA	1.12%	11.01%	1.11%	9.84%	0.96%	14.47%
	LTA	76.49%	80.66%	68.18%	71.54%	69.65%	**73.62%**
	Ours	**99.35%**	**99.98%**	**96.84%**	**94.96%**	**90.19%**	68.52%
Facebook	LSA	9.40%	43.55%	8.98%	34.45%	8.96%	38.63 %
	LTA	64.06%	62.92%	81.85%	80.41%	78.90%	77.63%
	Ours	**85.82%**	**86.50%**	**88.21%**	**91.94%**	**82.57%**	**98.88%**
Twitch	LSA	4.22%	37.65%	4.34%	36.52%	4.77%	46.02%
	LTA	56.66%	58.08%	66.95%	67.76%	31.61%	32.14 %
	Ours	**94.28%**	**96.11%**	**84.58%**	**96.68%**	**83.98%**	**53.41%**
LastFM	LSA	1.07%	25.00%	1.04%	25.43%	1.04%	22.80%
	LTA	32.65%	35.23%	41.31%	44.27%	23.37%	25.16%
	Ours	**99.04%**	**99.64%**	**93.32%**	**98.97%**	**80.66%**	**68.32%**

Results. We evaluate the attack accuracy as the fraction of labels correctly inferred. Table 2 shows the performance of label inference attacks. In most cases, our attack outperforms the two baselines. For instance, in the attack against GCN on the Citeseer dataset, Label-Infiltrator achieves an accuracy of 96.89%, while the attack accuracy of HG and MI is 75.56% and 94.61%, respectively. Note that HG can achieve an accuracy of up to 96.74%, which confirms that homophily can reflect the label information of the victim node. Benefiting from our infiltration strategy, MI outperforms HG in most cases. Gradient distance estimation can greatly promote the performance of label inference. For instance, in the attack against SAGE on Twitch, Label-Infiltrator boosts the accuracy of MI by nearly 30%.

4.2.2 Link-Infiltrator vs. Baselines. We compare the performance of Link-Infiltrator with two existing works.

- **Link Stealing Attack (LSA) [13].** LSA calculates the correlation distance between GNN posteriors of the node pairs. We collect distance scores between the victim node and each node in the candidate set. Then, we perform clustering on distance scores and identify the cluster with the lower average score as the neighborhood of the victim node. Note that LSA assumes that the adversary has access to the exact posteriors of all nodes, while our attack does not.
- **LinkTeller Attack (LTA) [47].** LTA performs influence analysis through the victim node and calculates influence scores of nodes in the candidate set. We then use density estimation for link estimation. Note that LTA assumes the adversary can manipulate attributes of the victim node, while our attack does not.

We compose a candidate set with a size of $|U| = 700$ for each victim node. The candidate set U includes all true neighbors of the victim node, i.e., $\mathcal{N}_v \subseteq U$, as well as other random non-neighbor nodes. U is set considering the size of commonly-used graph data. We also investigate the impact of the size of U in Section 4.3.5.

Table 4: Performance of attribute inference for continuous datasets. Cosine Distance and Euclidean Distance.

Dataset	GCN		SAGE		GAT	
	Cosine	Euclidean	Cosine	Euclidean	Cosine	Euclidean
Twitch	0.00	7.31E-05	0.00	1.93E-05	0.00	2.82E-05
LastFM	0.00	1.02E-05	0.00	9.49E-06	0.00	1.00E-05

Results. We use *Precision* and *Recall* to evaluate the attack performance. Precision quantifies the fraction of inferred neighbors that are true neighbors, and recall quantifies the fraction of true neighbors that are successfully detected. Table 3 shows the performance of link inference attacks. In most cases, our attack outperforms the two baselines. In the attack against GCN on the Cora dataset, Link-Infiltrator achieves nearly 100% recall, with only 1.61% false positive rate. LSA suffers from extremely low precision scores, indicating the clustering method includes too many false positives. LTA achieves a more stable performance. Note that Link-Infiltrator faces a slight decline on complex models, e.g., GAT, which is consistent with the conclusion in LTA [47].

4.2.3 Attribute-Infiltrator vs. Baselines. We compare Attribute-Infiltrator with two baselines.

- **Greedy Search (GS).** GS is a naive version of the Attribute-Infiltrator, where the adversary infiltrates directly to the victim, enumerates every possible value for the sensitive attributes, and preserves the value with the highest confidence score.
- **Homophily Guessing (HG).** According to the homophily theory [6, 25, 62], the attributes of neighboring nodes are also similar to each other. The baseline HG collects the values of the sensitive attributes from the neighbors of the victim node. The adversary then guesses the attribute values of the victim by majority vote. Note that HG assumes that the adversary has access to the attribute information of the neighbors of the victim nodes.

Results. We consider both discrete and continuous attributes. For discrete attributes, we evaluate the attack accuracy as the average fraction of attributes correctly inferred. As shown in Figure 3, our attack outperforms the two baselines on most datasets. For instance, Attribute-Infiltrator attains nearly 100% accuracy when inferring one sensitive attribute on the Facebook dataset and maintains a relatively high accuracy of 88.39% when there are 10 sensitive attributes. GS also benefits from our infiltration design and outperforms HG in most cases. We note that the attack accuracy decreases slightly when optimizing multiple sensitive attributes, but the dimension of non-zero attributes is relatively small in most graph datasets. For continuous attributes, we measure the performance by Cosine Distance and Euclidean Distance between the inferred attributes and the true attributes. As shown in Table 4, our attack has low distance values on Twitch and LastFM, which implies high accuracy of attribute inference.

4.2.4 Utility Loss. The infiltration operation of Infiltrator may affect the GNN model utility. A practical adversary should be

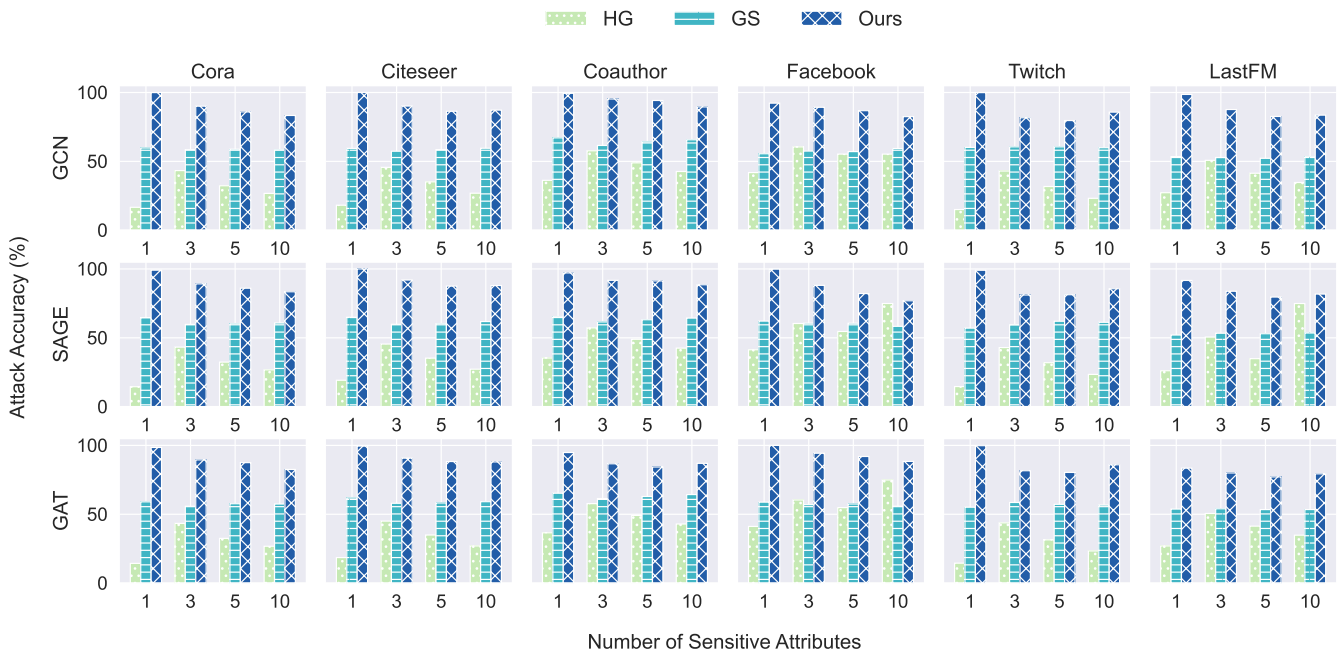

Figure 3: Attack performance of Attribute-Infiltrator. Different rows and columns represent different GNNs and datasets.

Table 5: Model utility under Infiltrator. Ori.: original model prediction accuracy. {Label-, Link-, Attribute-}: model utility under Infiltrator. c.a.: classification accuracy. c.c.: confidence score change of the true label.

Dataset	GNN	Ori.	Label-		Link-		Attribute-	
			c.a.	c.c.	c.a.	c.c.	c.a.	c.c.
Cora	GCN	99.36%	99.41%	-0.60	99.46%	-0.62	98.97%	-0.15
	SAGE	99.99%	99.95%	-1.35	99.95%	-1.35	99.95%	-1.32
	GAT	99.51%	99.56%	-1.88	99.56%	-1.91	99.46%	-1.93
Citeseer	GCN	96.75 %	97.35%	-0.35	97.15%	-1.95	96.15%	-4.07
	SAGE	99.99%	99.96%	-2.02	99.96%	-2.36	99.96%	-2.10
	GAT	96.71%	96.75%	-3.71	96.66%	-3.85	96.35%	-3.89
Coauthor	GCN	97.54 %	99.40%	-0.25	99.38%	-0.70	98.80%	-0.23
	SAGE	99.99%	99.99%	-0.76	99.99%	-0.82	99.99%	-0.80
	GAT	97.09%	98.96%	-2.19	98.91%	-2.35	98.86%	-2.24
Facebook	GCN	86.68%	96.71%	-0.59	96.82%	-0.56	96.44%	-0.40
	SAGE	91.22%	98.62%	-1.06	98.76%	-1.07	98.34%	-1.14
	GAT	88.26%	98.88%	-0.94	98.90%	-0.90	98.58%	-0.83
Twitch	GCN	87.12%	94.64%	+1.33	94.58%	+1.23	91.54%	+0.89
	SAGE	99.66%	99.80%	-0.86	99.80%	-0.87	99.80%	-0.87
	GAT	98.19%	98.02%	-2.13	97.99%	-2.16	97.99%	-2.24
LastFM	GCN	93.07%	97.85%	-1.85	97.57%	-2.90	96.45%	-2.76
	SAGE	95.33%	96.78%	-6.68	96.71%	-7.17	96.40%	-7.27
	GAT	98.67%	98.08%	-2.95	97.76%	-3.49	98.04%	-3.00

stealthy to launch attacks without degrading model classification accuracy on victim nodes. Otherwise, the attack may be detected. We use two metrics to measure the model utility under Infiltrator: (1) *classification accuracy* of the victim node under Infiltrator, and (2) the *relative change* of the confidence score of the true label of the victim node.

Results. As shown in Table 5, Infiltrator only slightly degrades or even enhances GNN model utility. According to the relative change of confidence score, Infiltrator has little effect on the GNN output. For instance, the worst case is the Attribute-Infiltrator on the SAGE model and LastFM dataset, where the classification accuracy only declines 7.27%.

4.3 RQ2: Robustness of Infiltrator

To deploy Infiltrator, we first consider several non-ideal attack conditions (Sections 4.3.1-4.3.3). We also show that Infiltrator is robust to various attack setups (Sections 4.3.4-4.3.6) and is efficient for query limits (Section 4.3.7).

4.3.1 Impact of Model Overfitting on Label-Infiltrator. Recent studies have discussed the relationship between model overfitting and privacy leakage of machine learning models [3, 32]. Empirical results show that overfitted models are more vulnerable, e.g., under membership inference attacks [32]. We investigate the effectiveness of Infiltrator under varying overfitting levels, which is quantified by the training and testing accuracy gap of the GNN [14, 32]. We control the number of training epochs to achieve different overfitting levels. To attain a relatively high overfitting level, we train GNNs for 200 epochs. To attain a low overfitting level, we employ early stopping, weight decay, and dropout mechanisms to reduce overfitting.

Results. We demonstrate the relationship between the overfitting level and the attack accuracy in Table 6. Note that the overfitting level cannot be precisely controlled, thus we compare the results of relatively-high and relatively-low overfitting levels. It is surprising that the attack accuracy of Infiltrator is comparable and even higher when the overfitting level is lower, which is

Table 6: Impact of model overfitting on Label-Infiltrator. o.f.: overfitting level. acc.: attack accuracy.

		GCN		SAGE		GAT	
Coauthor	o.f.	0.64%	3.34%	0.82%	4.14%	0.84%	4.67%
	acc.	99.11%	99.30%	98.77%	98.45%	97.22%	96.54%
Coauthor partial	o.f.	0.96%	7.77%	0.98%	5.18%	1.07%	6.34%
	acc.	99.67%	99.59%	98.83%	96.34%	96.75%	97.35%

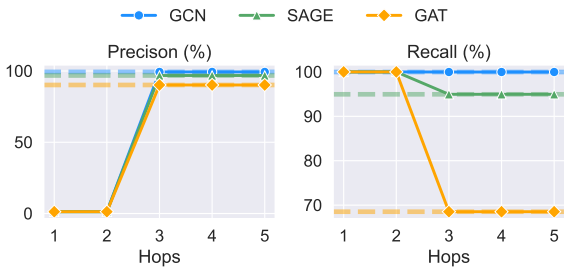

Figure 4: Impact of graph evolution on Link-Infiltrator on Coauthor.

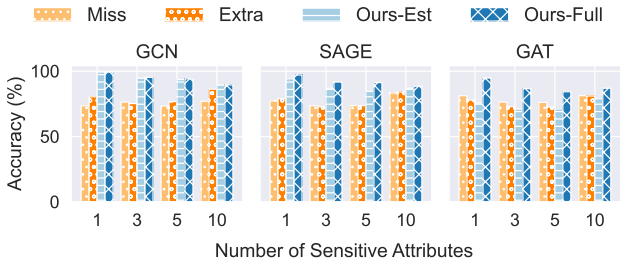

Figure 5: Impact of prior knowledge on Attribute-Infiltrator on Coauthor.

Table 7: Impact of infiltrator nodes attributes. acc.: attack accuracy. pre.: precision. rec.: recall.

Citeseer		GCN		SAGE		GAT	
		nil	random	nil	random	nil	random
Label	acc.	96.89%	95.52%	81.24%	80.72%	92.66%	95.84%
Link	pre.	98.27%	99.89%	99.39%	95.44%	91.88%	96.99%
	rec.	99.84%	99.83%	99.75%	94.87%	93.35%	94.83%

different from previous privacy attacks [14, 32, 36]. The possible reason is that our attack does not rely on the *memorization* of the GNN (which relates to the overfitting level) but the *behavior* of the GNN. More specifically, our inference is derived from the similarity between the behavior of the GNN on the victim node and the infiltrator node. We further compare the results of Coauthor with a smaller (25% partial) subgraph of Coauthor, and results demonstrate that Label-Infiltrator is not sensitive to the graph scale.

4.3.2 Impact of Graph Evolution on Link-Infiltrator. In the real world, graphs may evolve with time, e.g., social networks. We analyze whether graph evolution affects the attack performance of Link-Infiltrator since Link-Infiltrator is more sensitive to nodes/links changes due to the message passing mechanism of GNNs. Recall that Link-Infiltrator performs two-step infiltration. We simulate graph evolution by injecting a new node into the graph after the adversary gets $p_{a,0}$ and before the adversary obtains $p_{a,u}$ for link inference. We consider the new node is at different hops from the victim node v.

Results. As shown in Figure 4, a new node injected more than 2 hops away from the victim node will not affect the attack performance of Link-Infiltrator. If a new node is injected within 2 hops from the victim node, $p_{a,u}$ will be diverted from $p_{a,0}$, which leads to a steep decline in the attack precision. Note that the adversary can mitigate this potential problem by adapting the inference scheme through monitoring $p_{a,0}$, with details in Appendix E.1.

4.3.3 Impact of Prior Knowledge on Attribute-Infiltrator. Note that Attribute-Infiltrator may require the prior knowledge of the neighbor information of the victim node. In the previous experiments, we assume that the adversary knows all the real neighbors $\mathcal{N}_v$ of the victim node. In this part, we analyze the effectiveness of Attribute-Infiltrator under partial or even faulty knowledge. We consider two possible average cases where the adversary misses 20% of real neighbors (denoted as Miss) or includes 20% extra fake neighbors (denoted as Extra) when performing infiltration. We also perform an end-to-end attribute inference with Link-Infiltrator and Attribute-Infiltrator (denoted as Ours-Est). We then compare their attack effectiveness with Attribute-Infiltrator with full knowledge of the neighbors of the victim (denoted as Ours-Full).

Results. As shown in Figure 5, Attribute-Infiltrator with missing real neighbors and extra fake neighbors can still achieve relatively high attack accuracy, outperforming baselines. Due to the high accuracy of Link-Infiltrator, Ours-Est attains comparable performance with Ours-Full. Even with a worse estimation against GAT (with 68.52% recall), Ours-Est can achieve over 70% accuracy. According to the worst case of link inference accuracy in Table 3, we consider up to 50% more/fewer neighbors in Appendix D.1.

4.3.4 Impact of Infiltrator Nodes Attributes. We preset the attribute vectors of the infiltrator nodes as $\mathbf{0}$, which may not be feasible in some cases, e.g., social networks may not allow creating blank accounts. Therefore, we evaluate the performance of Infiltrator when the attribute vectors of the infiltrator nodes are assigned as random values.

Results. As shown in Table 7, both Label-Infiltrator and Link-Infiltrator are robust to the change of the default nil attributes of infiltrator nodes. We omit Attribute-Infiltrator as it relies on the iterative optimization of the attributes.

4.3.5 Impact of Candidate Set Size on Link-Infiltrator. Considering the scale of commonly-used graph data, we craft the candidate set U with the size of 700. For instance, the Facebook dataset has the largest average number of neighbors (~64), and the Citeseer dataset has the smallest average number of neighbors (~3). To avoid missing true neighbors, it is best for an attacker to consider the

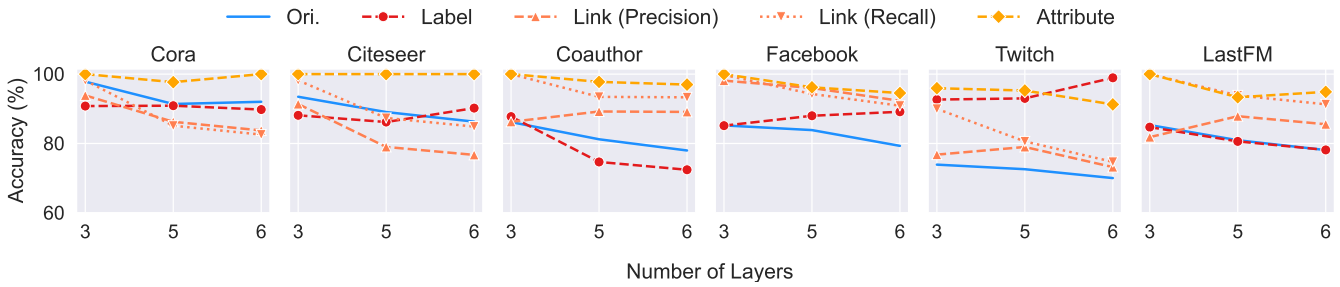

Figure 6: Scalability of Infiltrator against deeper GINs. Ori.: original prediction accuracy of GINs.

Table 8: Link-Infiltrator with different sizes of candidate set U. The entire graph of Citeseer is composed of 3,327 nodes.

Citeseer	$\|U\| = 700$		$\|U\| = 2,000$		Entire graph	
	pre.	rec.	pre.	rec.	pre.	rec.
GCN	98.27%	99.84%	99.46%	99.83%	99.46%	99.83%
SAGE	99.39%	99.75%	95.78%	99.65%	92.66%	99.65%
GAT	91.88%	93.35%	96.91%	93.35%	96.51%	94.63%

entire graph as the candidate set U, which naturally increases computational overhead. In addition, the increase of non-neighbors in U may incur false positives, i.e., non-neighbors inferred as neighbors.

Results. In Table 8, we show the performance under different sizes of the candidate set. We test on Citeseer, as with the smallest average number of neighbors, precisely inferring neighbors can be non-trivial. It is shown that Link-Infiltrator is robust to the size of the candidate set. Without loss of generality, we use the default $|U| = 700$ with all true neighbors incorporated.

4.3.6 Scalability of Infiltrator (Impact of Model Complexity). We analyze the scalability of Infiltrator against more complex GNNs. We here define the GNN complexity as the number of hidden layers. In previous studies [45, 47], GNNs with deeper layers suffer from poor classification results (e.g., nearly 25% decline on 5-layer GATs), thus interfering with the evaluation. Therefore, we introduce an advanced GNN, namely Graph Isomorphism Networks (GIN) [50]. GIN is initially proposed and tested with deeper layers and achieves satisfactory performance (with details in Appendix A). In this part, we assess the performance of Infiltrator against more complex GINs with hidden layers varying from $\{3, 5, 6\}$.

Results. As shown in Figure 6, with deeper layers, the prediction accuracy of GINs decreases, which also affects the performance of Label-Infiltrator, e.g., on LastFM, the attack accuracy closely follows the model prediction accuracy. The performance of Link-Infiltrator also deteriorates since the adversary may capture redundant influence information from non-neighbors as the GNN deepens, but the attack accuracy is still over 70%. We present Attribute-Infiltrator on one sensitive attribute, which maintains comparable attack performance. Overall, we demonstrate that Infiltrator is scalable to deeper (and well-performed) GNNs.

4.3.7 Query Costs. We analyze the query costs of Infiltrator to show that Infiltrator is query-efficient.

- **Label-Infiltrator.** For label inference, Algorithm 1 performs $(1+2\cdot|C|)$ queries. In a general case where we consider the top-2 classes for gradient estimation, i.e., $|C| = 2$, Label-Infiltrator only requires 5 model queries.
- **Link-Infiltrator.** Link-Infiltrator necessarily requires $(1+|U|)$ queries to determine all neighbors among $|U|$ candidates, i.e., to obtain only one posterior for each node in U.
- **Attribute-Infiltrator.** The query costs depend on the optimization iterations in Algorithm 3. As discussed in Appendix E.2, we set at most 300 queries for 10 attributes inference. Note that this can be overmuch for fewer attributes, e.g., we empirically set 4 for a single discrete attribute.

4.4 RQ3: Transferability of Infiltrator

Inductive GNNs are able to predict unseen data without additional re-training. Therefore, an inductive GNN trained on one graph may be used to perform predictions on (1) the evolving graph or (2) another new graph. We evaluate the transferability of Infiltrator in these two inductive settings.

4.4.1 Attacks on Inductive Nodes. In this setting, we consider attacks on non-training nodes on the same graph as the victim nodes. For Attribute-Infiltrator, we focus on the results of the single attribute inference for illustration. As shown in Figure 7, our attacks on inductive nodes attain a comparable performance with training nodes. The possible reason is that well-trained GNNs have a high prediction accuracy on both training and inductive nodes. The high attack performance on inductive nodes indicates that Infiltrator can also successfully mount attacks without knowing the victim's membership. We supplement attack results of multiple sensitive attributes in Figure 8.

4.4.2 Attacks on Inductive Graphs. In this setting, we consider attacks on non-training nodes on a totally different graph as the victim nodes. We conduct experiments on the Twitch [28] datasets. Twitch consists of various social networks collected from different countries, while all graphs have the same set of node attributes. We select three graphs, namely Twitch-{ES, DE, PT} (with details in Appendix D.2). Figure 9 illustrates the attack results of Infiltrator on various inductive task pairs, e.g, GNN models are pre-trained on nodes in Twitch-ES and applied on nodes in Twitch-DE. Infiltrator achieves comparable performance across datasets and GNN models. The possible reason is that GNNs are able to predict unseen graphs sampled from the same distribution of the training graph.

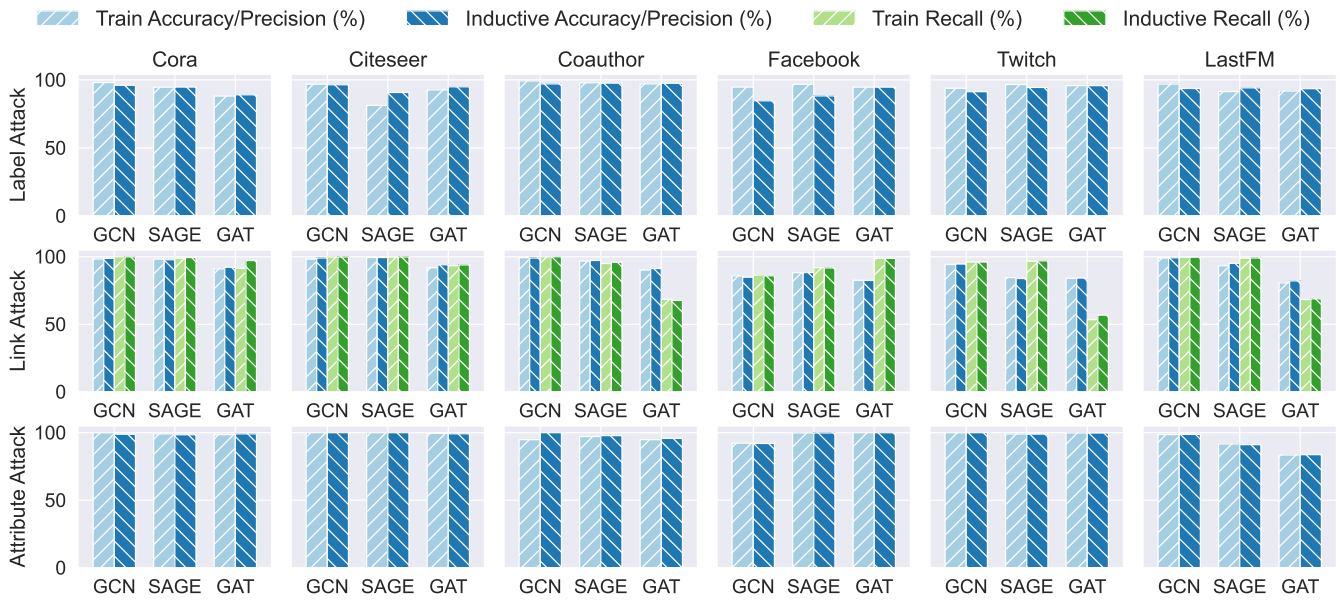

Figure 7: Transferability of Label-, Link-, and Attribute-Infiltrator (one sensitive attribute) on inductive nodes as compared with that of training nodes.

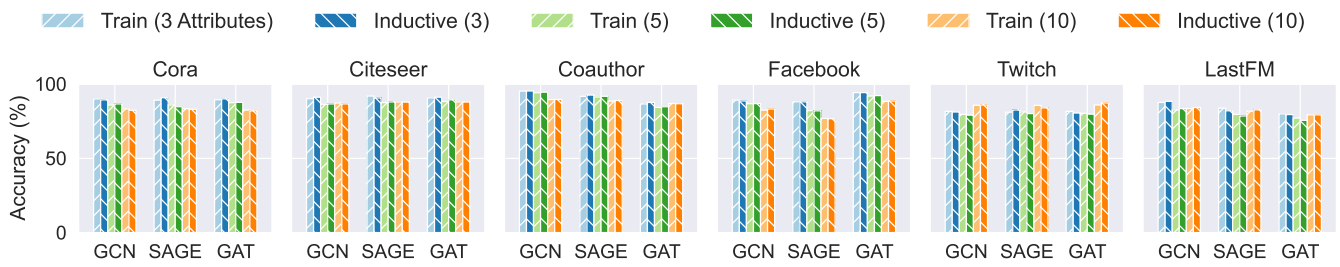

Figure 8: Transferability of Attribute-Infiltrator (multiple sensitive attributes) on inductive nodes as compared with that of training nodes.

The high attack performance on inductive graphs indicates that Infiltrator is also capable of attacking unseen graphs.

5 DEFENSES

In this section, we evaluate the resistance of Infiltrator against possible defenses. In particular, we consider two possible defenses, i.e., (1) the graph homophily defender, and (2) the differentially private (perturbation-based) GNN.

5.1 Graph Homophily Analysis

Existing works [25] realize that graphs, especially social networks, show strong homophily. Homophily-based defenses [6, 57] have been devised to detect attacks against graphs [64, 65]. The key insight is that the modification or injection of adversarial nodes and edges will deteriorate the homophily distribution.

We perform homophily analyses for all datasets before and after the infiltration. We adopt two metrics of homophily defined in [6]. (1) Node-centric homophily compares the similarity between the target node and its neighbors. (2) Edge-centric homophily compares the similarity between the nodes connected by a target edge. Detailed definitions and explanations are in Appendix C.1.

Results. We illustrate the homophily distribution in Figure 10. Since we adopt cosine similarity for homophily analyses, the distribution of discrete and continuous datasets range from $[0, 1]$ and $[-1, 1]$, respectively. The homophily defenders detect the malicious node or edges with abnormally lower homophily since adversarial nodes are often crafted to be different from the original nodes. However, as shown in Figure 10, we cannot find an apparent deviation after Infiltrator attacks. This is because the perturbation scale of the infiltration set of Infiltrator is much smaller than that of adversarial attacks [64]. Moreover, Figure 10 shows that (1) social networks have higher homophily (with the median value highlighted) than citation networks, and (2) the node-centric homophily captures higher similarities than edge-centric homophily.

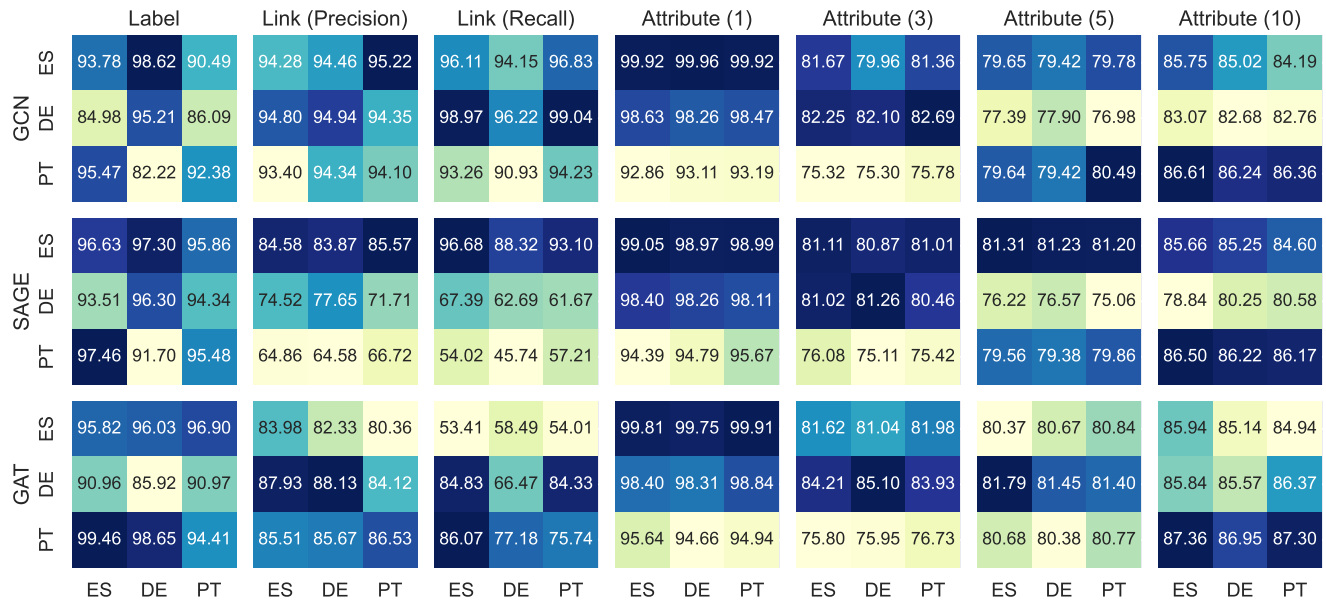

Figure 9: Transferability of Infiltrator on inductive graphs. Different rows and columns are for different models and attacks. Datasets Twitch-{ES, DE, PT} on the vertical axis are for GNN training, and on the horizontal axis are for Infiltrator attacks.

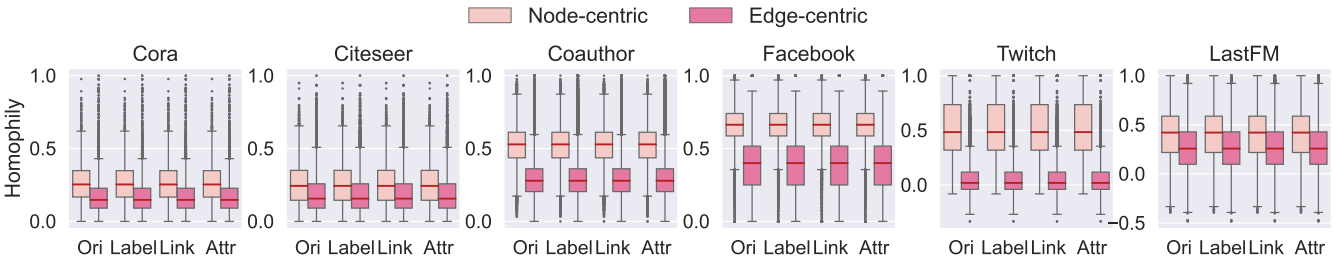

Figure 10: Homophily distributions under Infiltrator. Ori.: original homophily distribution. {Label, Link, Attr}: homophily distributions under Label-, Link-, and Attribute-Infiltrator.

Such results are consistent with the statement in previous studies [6, 25]. Experiment results show that Infiltrator has good resistance against homophily defenders.

5.2 Differentially Private GNNs

Differentially Private (DP) GNNs [23, 31, 47] have been designed for protecting sensitive graph data. We mainly evaluate the resistance of Infiltrator against a representative DP GNN and two perturbation-based mechanisms.

5.2.1 GAP. We choose the state-of-the-art GAP (DP GNNs with Aggregation Perturbation) [31] for evaluation. While other models provide edge-level privacy, GAP can also provide node-level privacy guarantees. Specifically, GAP considers not only links (neighbors) but also attributes and labels that are private for graph nodes. We implement the GAP variants without DP and with node-level DP, denoted as GAP-INF and GAP-NDP, respectively. The key parameter of the DP mechanism is the privacy budget ϵ. A lower privacy budget leads to a stronger privacy guarantee but reduced utility of GAP models. For GAP-NDP, we set $\epsilon = 8$, whose model shows comparable model utility and defense effectiveness in extensive experiments of [31]. For non-private GAP-INF, $\epsilon = \infty$. For other GAP settings, we follow [31] and elaborate the details in Appendix C.2.

Results. As shown in Table 9, our test accuracy is consistent with the original work [31], with only ~7.6% overfitting level. Compared to the results on conventional GNNs, Infiltrator achieves comparable attack performance on GAP-INF. For instance, the accuracy of Attribute-Infiltrator against GAP-INF is nearly 100%, which is even higher than that against GCN. This indicates that Infiltrator fits well on the novel structure of GAP, which does not follow the ConvGNN design. However, we also notice the steep degradation when mounting Link- and Attribute-Infiltrator on GAP-NDP. However, GAP-NDP suffers from the deteriorated model utility. For instance, its classification accuracy on testing nodes decreases by nearly 50% (48.51%) on the Cora dataset compared with

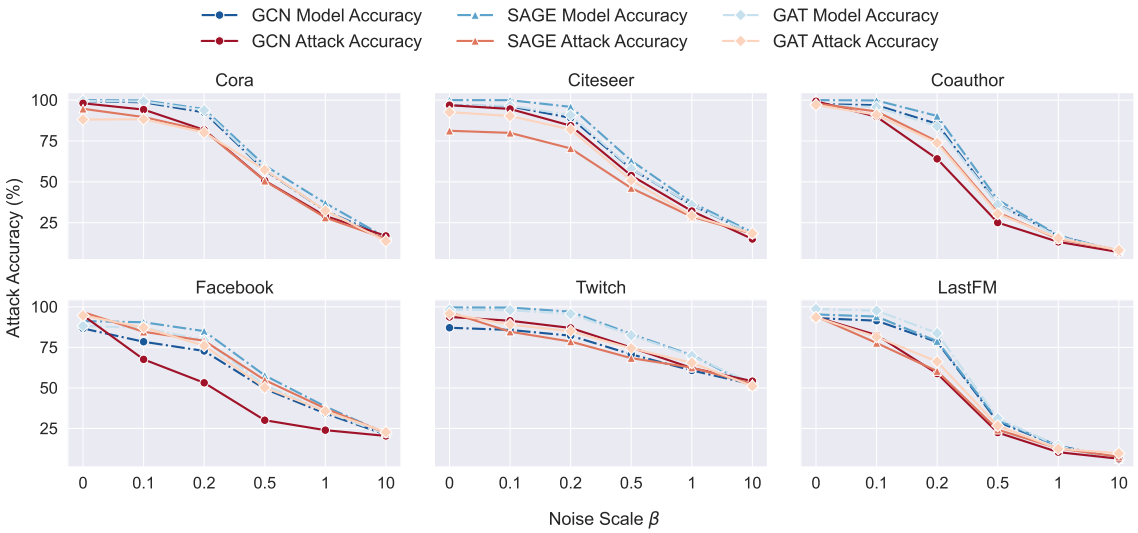

Figure 11: Label-Infiltrator against Laplace noise (scaled by β). We illustrate both attack accuracy (attack effectiveness) and training accuracy (model utility) on different datasets and GNN models.

Table 9: Attack performance of Infiltrator against GAP. Ori.: prediction accuracy of GAP models.

Dataset	GAP	Ori.	Label-	Link-		Attribute-
				pre.	rec.	
Cora	INF	77.10%	95.25%	100.00%	100.00%	100.00%
	NDP	28.59%	100.00%	0.60%	50.40%	67.93%
Citeseer	INF	77.20%	98.75%	100.00%	100.00%	100.00%
	NDP	32.07%	82.64%	3.80%	52.50%	61.52%
Coauthor	INF	95.03%	99.76%	100.00%	100.00%	99.97%
	NDP	64.63%	98.45%	0.23%	47.93%	59.00%
Facebook	INF	84.82%	91.51%	100.00%	99.99%	100.00%
	NDP	63.53%	75.38%	8.57%	41.03%	67.75%
Twitch	INF	71.00	100.00%	100.00%	100.00%	99.80%
	NDP	58.61%	100.00%	3.55%	49.06%	59.91%
LastFM	INF	79.85%	94.23%	100.00%	100.00%	99.66 %
	NDP	40.45%	98.16%	1.09%	49.70%	52.89%

that of GAP-INF. As such, we only conduct Label-Infiltrator on correctly classified nodes for fair evaluation, which maintains comparable attack accuracy.

5.2.2 Laplacian Perturbation. A common method of perturbation-based privacy defenses is imposing noise on data or models, which may deteriorate the information exposed to adversaries and is shown to be effective against previous inference attacks [45, 58]. Imposed on the posteriors, the Laplace perturbation will affect not only Infiltrator attacks but also GNN predictions. We elaborate the detailed settings of the Laplacian noise in Appendix C.3. We analyze the trade-off between the model utility and the attack effectiveness of Label-Infiltrator, characterized by model accuracy and attack accuracy. We evaluate the defense with different noise scales $\beta = \{0.1, 0.2, 0.5, 1, 10\}$.

Results. As shown in Figure 11, we observe that the increasing noise scale will degrade attack accuracy since the large-scale noise confuses Label-Infiltrator from both confidence score inference and gradient estimation inference (in Algorithm 1). Moreover, the dropping attack accuracy is due to the decreasing GNN performance, which Label-Infiltrator largely relies on. However, the model accuracy also significantly drops, which may harm the utility of the model.

5.2.3 Adversarial Training. In this part, we evaluate the training-time defense against Infiltrator. We show that introducing training noise is less effective than previous inference-time defenses, which can directly interfere with GNN outputs.

We adopt Adversarial Training (AT) [11] to enhance model training. The insight to use AT for Infiltrator is that AT can lead to robust GNN posteriors, which may impede inference. Here, we impose the perturbation on node attributes to evaluate the potential impact on Attribute-Infiltrator (formulated in Appendix C.4).

Results. We retrain GCNs with GraphAT [11], which incorporates the dynamic regularizer in the objective function of GNN training. With a slight decline of model accuracy under GraphAT, Table 10 shows that Attribute-Infiltrator performs even better under AT. Existing work [37] also indicates that adversarially robust models may even increase privacy risks.

6 RELATED WORK

Injection attacks against GNNs. Extensive studies show that GNNs are vulnerable to adversarial attacks [18, 38, 49]. Recent studies [39, 43, 64] (including a KDD-CUP competition[9]) put forward Graph Injection Attacks (GIA). However, early attacks [39, 43, 44] focus on poisoning scenarios that require additional GNN retraining.

[9] https://biendata.xyz/competition/kddcup_2020_formal

Table 10: Attack accuracy of Attribute-Infiltrator against GraphAT. Ori.: original model accuracy.

Dataset	Ori.	Dimension of Target Attribute(s)			
		1	3	5	10
Cora	94.04%	99.95%	91.99%	88.68%	84.64%
Citeseer	86.07%	100.00%	92.32%	88.70%	89.10%
Coauthor	91.50%	99.75%	95.82%	96.37%	89.80%
Facebook	83.99%	100.00%	96.75%	95.98%	93.38%
Twitch	70.37%	91.07%	81.07%	84.34%	89.80%
LastFM	86.17%	99.88%	91.86%	89.22%	87.90%

For inference stage attacks, Tao et al. [40] proposed the single node injection. Zou et al. [64] located the topological defective nodes suitable for injection with a surrogate model under the black-box setting. Wang et al. [46] approximated adversarial vulnerabilities of victim nodes to perform the clustering-based injection. Chen et al. [6] further improved the defense resilience of GIA. Different from GIAs that aim to induce misclassification of the victim node, Infiltrator is devoted to inferring private information of the victim node.

Privacy attacks against GNNs. Recent studies explore the privacy leakage of graph data through GNN models. Pioneering studies attempt to infer the training membership. Duddu et al. [9] exploited the statistical difference in posterior distribution between training and testing nodes. He et al. [14] obtained 0-hop and 2-hop posterior information to train an attack classifier. Conti et al. [7] extended the attack to a label-only setting assisted with auxiliary features. A variety of attacks are proposed for different sensitive information. Zhang et al. [58] discovered the graph-level property inference and subgraph-level membership inference. Wang et al. [45] developed the group-level property inference. Some studies focus on inferring graph links. He et al. [13] proposed the link-stealing attack by comparing the similarity between two given nodes. Wu et al. [47] performed link inference through posterior changes before and after feature manipulations. More aggressive attacks [9, 34, 58, 60] aimed to reconstruct the complete link information of the graph. Duddu et al. [9] also consider the attribute inference attack for graph nodes but target the random-walk-based embeddings rather than GNN-based outputs. We distinguish Infiltrator from these studies as (1) we focus on fundamental node-level data information and (2) we consider a strict black-box setting with only the posteriors of certain nodes from GNNs.

Privacy-preserving GNNs. Differential privacy has been widely applied to design privacy-preserving machine learning models [1, 16]. In the context of graph data, early studies [19, 20] investigate the edge-level and node-level DP. Sajadmanesh et al. [30] propose locally private GNN, which can handle DP noisy features in data partition settings. Wu et al. [47] propose practical DP training methods via graph structure input perturbation, which provides GCNs with edge-level privacy. Kolluri et al. [23] design a link private model with novel MLP-based structures while attaining better privacy-utility trade-offs. Node-level privacy is harder to attain since it protects all information pertaining to graph nodes (including their linked edges). Daigavane et al. [8] adapts the DP-SGD (Stochastic Gradient Descent) to train node-level DP GNNs. Ayle et al. [2] generate disjoint subgraphs for node feature protection. Very recently, Sajadmanesh et al. [31] have proposed GAP, a novel network that can provide both edge- and node-level DP guarantees. In this paper, we take GAP as the state-of-the-art privacy-preserving model to evaluate the defense resistance of Infiltrator.

7 CONCLUSION

In this paper, we investigate the node-level privacy leakage of the burgeoning Graph Neural Networks. We propose Infiltrator, an inference attack framework leveraging adversarial infiltration. Infiltrator can estimate the private label, neighboring links, and sensitive attributes of the victim node through minimal infiltration. Extensive experiments have verified the attack effectiveness and defense resistance of Infiltrator.

ACKNOWLEDGMENTS

This work was supported by Ant Group.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *ACM SIGSAC conference on computer and communications security*. 308–318.
[2] Morgane Ayle, Jan Schuchardt, Lukas Gosch, Daniel Zügner, and Stephan Günnemann. 2023. Training differentially private graph neural networks with random walk sampling. *arXiv preprint arXiv:2301.00738* (2023).
[3] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy*. 1897–1914.
[4] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Security*. 15–26.
[5] Weijian Chen, Yulong Gu, Zhaochun Ren, Xiangnan He, Hongtao Xie, Tong Guo, Dawei Yin, and Yongdong Zhang. 2019. Semi-supervised user profiling with heterogeneous graph attention networks. In *International Joint Conference on Artificial Intelligence*. ijcai.org, 2116–2122.
[6] Yongqiang Chen, Han Yang, Yonggang Zhang, Kaili Ma, Tongliang Liu, Bo Han, and James Cheng. 2022. Understanding and improving graph injection attack by promoting unnoticeability. In *International Conference on Learning Representations*. OpenReview.net.
[7] Mauro Conti, Jiaxin Li, Stjepan Picek, and Jing Xu. 2022. Label-only membership inference attack against node-Level graph neural networks. In *ACM Workshop on Artificial Intelligence and Security*. 1–12.
[8] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2021. Node-level differentially private graph neural networks. *arXiv preprint arXiv:2111.15521* (2021).
[9] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying privacy leakage in graph embedding. In *EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 76–85.
[10] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. ACM, 417–426.
[11] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. 2019. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2019), 2493–2504.
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 1024–1034.
[13] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing links from graph neural networks. In *USENIX Security Symposium*. 2669–2686.
[14] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. 2021. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429* (2021).
[15] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *AAAI Conference on Artificial Intelligence*. AAAI Press, 4115–4122.

[16] Bargav Jayaraman and David Evans. 2019. Evaluating differentially private machine learning in practice. In *USENIX Security Symposium*. 1895–1912.

[17] Bargav Jayaraman and David Evans. 2022. Are attribute inference attacks just imputation?. In *ACM SIGSAC Conference on Computer and Communications Security*. 1569–1582.

[18] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. 2021. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 19–34.

[19] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.

[20] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*. Springer, 457–476.

[21] Nicolas Keriven and Gabriel Peyré. 2019. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 7090–7099.

[22] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*. OpenReview.net.

[23] Aashish Kolluri, Teodora Baluta, Bryan Hooi, and Prateek Saxena. 2022. LPGNet: Link private graph networks for node classification. In *ACM SIGSAC Conference on Computer and Communications Security*. 1813–1827.

[24] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1569–1572.

[25] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001), 415–444.

[26] Shagufta Mehnaz, Sayanton V Dibbo, Roberta De Viti, Ehsanul Kabir, Björn B Brandenburg, Stefan Mangard, Ninghui Li, Elisa Bertino, Michael Backes, Emiliano De Cristofaro, et al. 2022. Are your sensitive attributes private? Novel model inversion attribute inference attacks on classification models. In *USENIX Security Symposium*. 4579–4596.

[27] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*. OpenReview.net.

[28] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale attributed node embedding. *arXiv preprint arXiv:1909.13021* (2019).

[29] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *ACM International Conference on Information and Knowledge Management*. 1325–1334.

[30] Sina Sajadmanesh and Daniel Gatica-Perez. 2021. Locally private graph neural networks. In *ACM SIGSAC Conference on Computer and Communications Security*. 2130–2145.

[31] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. 2022. GAP: Differentially private graph neural networks with aggregation perturbation. *arXiv preprint arXiv:2203.00949* (2022).

[32] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Network and Distributed System Security Symposium*. The Internet Society.

[33] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[34] Yun Shen, Yufei Han, Zhikun Zhang, Min Chen, Ting Yu, Michael Backes, Yang Zhang, and Gianluca Stringhini. 2022. Finding MNEMON: Reviving memories of node embeddings. In *ACM SIGSAC Conference on Computer and Communications Security*. 2643–2657.

[35] Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. 2022. Model stealing attacks against inductive graph neural networks. In *IEEE Symposium on Security and Privacy*. 1175–1192.

[36] Congzheng Song and Vitaly Shmatikov. 2020. Overlearning reveals sensitive attributes. In *International Conference on Learning Representations*. OpenReview.net.

[37] Liwei Song, Reza Shokri, and Prateek Mittal. 2019. Privacy risks of securing machine learning models against adversarial examples. In *ACM SIGSAC Conference on Computer and Communications Security*. 241–257.

[38] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, S Yu Philip, Lifang He, and Bo Li. 2022. Adversarial attack and defense on graph data: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[39] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. 2020. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *The Web Conference*. ACM / IW3C2, 673–683.

[40] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. 2021. Single node injection attack against graph neural networks. In *ACM International Conference on Information and Knowledge Management*. 1794–1803.

[41] Amanda L Traud, Peter J Mucha, and Mason A Porter. 2012. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications* 391, 16 (2012), 4165–4180.

[42] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*. OpenReview.net.

[43] Jihong Wang, Minnan Luo, Fnu Suya, Jundong Li, Zijiang Yang, and Qinghua Zheng. 2020. Scalable attack on graph data by injecting vicious nodes. *Data Mining and Knowledge Discovery* 34, 5 (2020), 1363–1389.

[44] Xiaoyun Wang, Minhao Cheng, Joe Eaton, Cho-Jui Hsieh, and Felix Wu. 2018. Attack graph convolutional networks by adding fake nodes. *arXiv preprint arXiv:1810.10751* (2018).

[45] Xiuling Wang and Wendy Hui Wang. 2022. Group property inference attacks against graph neural networks. In *ACM SIGSAC Conference on Computer and Communications Security*. 2871–2884.

[46] Zhengyi Wang, Zhongkai Hao, Ziqiao Wang, Hang Su, and Jun Zhu. 2022. Cluster Attack: Query-based adversarial attacks on graph with graph-dependent priors. In *International Joint Conference on Artificial Intelligence*. ijcai.org, 768–775.

[47] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2022. LinkTeller: Recovering private edges from graph neural networks via influence analysis. In *IEEE Symposium on Security and Privacy*. 2005–2024.

[48] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[49] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* 17, 2 (2020), 151–178.

[50] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *International Conference on Learning Representations*. OpenReview.net.

[51] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*. PMLR, 40–48.

[52] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. 2019. Neural network inversion in adversarial setting via background knowledge alignment. In *ACM SIGSAC Conference on Computer and Communications Security*. 225–240.

[53] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 4805–4815.

[54] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 5171–5181.

[55] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence*. AAAI Press, 4438–4445.

[56] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. 2021. MagNet: A neural network for directed graphs. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 27003–27015.

[57] Xiang Zhang and Marinka Zitnik. 2020. GNNGuard: Defending graph neural networks against adversarial attacks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 9263–9275.

[58] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. 2022. Inference attacks against graph neural networks. In *USENIX Security Symposium*. 1–18.

[59] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2022. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* 34, 1 (2022), 249–270.

[60] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. 2021. GraphMI: Extracting private graph data from graph neural networks. In *International Joint Conference on Artificial Intelligence*. ijcai.org, 3749–3755.

[61] Yadi Zhou, Fei Wang, Jian Tang, Ruth Nussinov, and Feixiong Cheng. 2020. Artificial intelligence in COVID-19 drug repurposing. *The Lancet Digital Health* 2, 12 (2020), e667–e676.

[62] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 7793–7804.

[63] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* 34, 13 (2018), i457–i466.

[64] Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. 2021. TDGIA: Effective injection attacks on graph neural networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2461–2471.

[65] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2847–2856.