Exploring Backdoors in Federated Graph Neural Networks

Jing Xu [1], **Stefanos Koffas [1]**, Stjepan Picek [1, 2]

[1]Deflt University of Technology

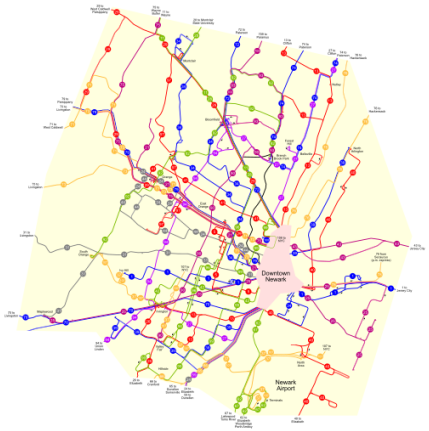
[2]Radboud University

December 6, 2022

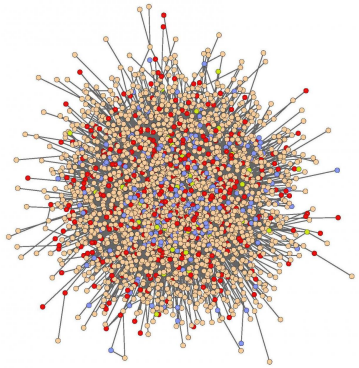
Graph Neural Netowrks (GNN)

Networks are "everywhere"

- Physical networks



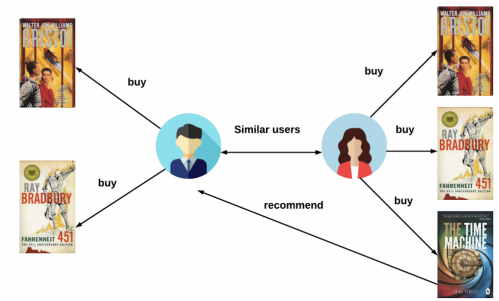
(a) Transportation Network



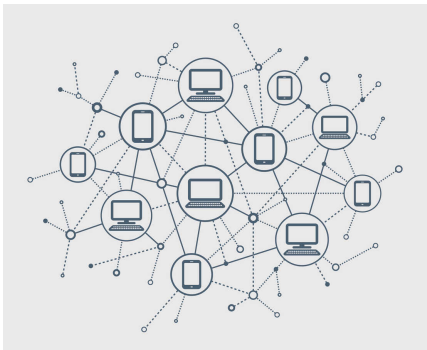
(b) Molecular Network

- Model complex relationships

(c) Social Network



(d) User-Item Network



(e) Web Network

Graph Neural Networks

Graph Neural Network is a type of Neural Network which directly operates on the graph structure.

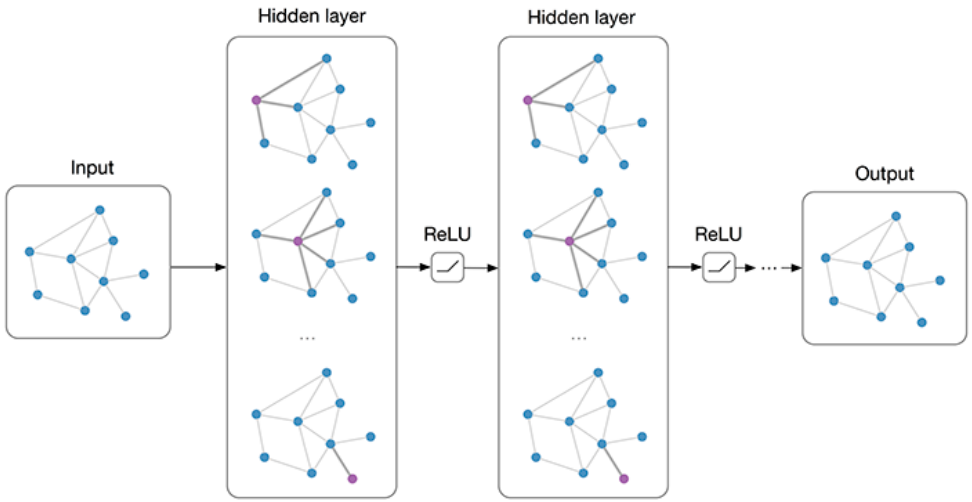


Figure 1: Multi-layer Graph Convolutional Network (GCN).

- The input is a graph including an adjacency matrix and a feature matrix for all nodes.
- GNNs learn features about the graph's structure and their nodes.
- GNNs are used in both graph and node classification tasks.

Federated Learning (FL)

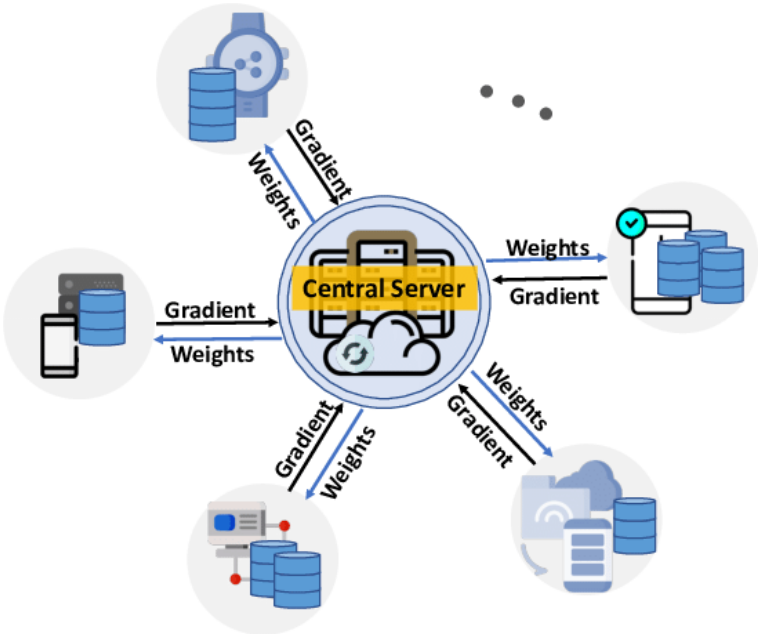


Figure 2: Federated learning framework.

- A distributed learning paradigm enabling multiple clients to train a global model collaboratively without revealing local data.
- Ensures data privacy, data security, data access rights and allows usage of heterogeneous data.
- Cross-device setting (e.g., android keyboard), cross-silo setting (e.g., drug discovery from different pharmaceutical institutions).

Federated GNNs

- Due to privacy concerns and regular restrictions, centralized GNNs can be difficult to apply to data-sensitive scenarios (e.g., when pharmaceutical institutions want to collaborate for drug discovery but cannot share their data)
- FL is a promising solution for training GNNs over isolated graph data, and there are already some works utilizing FL to train GNNs[1][2] [3], which we denote as Federated GNNs.
- We assume a cross-silo setting, so up to 100 clients is realistic [4].

[1] "Fedgraphnn: A federated learning system and benchmark for graph neural networks" (2021). In: *arXiv*

[2] "Peer-to-peer federated learning on graphs" (2019). In: *arXiv*

[3] "Federated Graph Learning–A Position Paper" (2021). In: *arXiv*

[4] "Advances and open problems in federated learning" (2021). In: *Foundations and Trends® in Machine Learning* 14

Backdoor Attacks

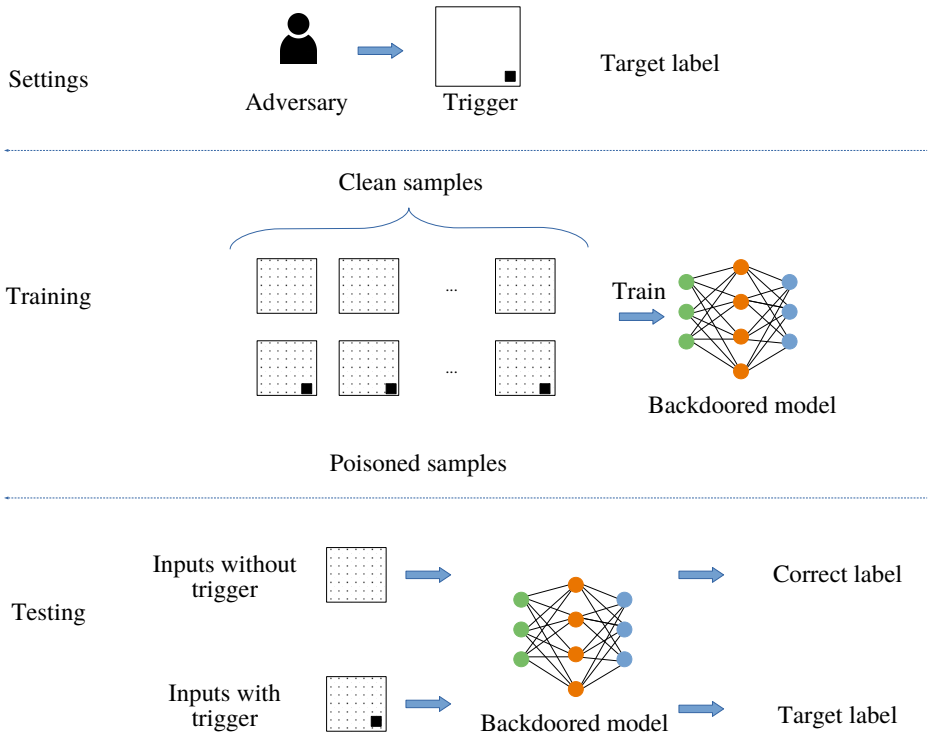


Figure 3: Backdoor attack framework.

- Backdoor attacks aim to make a model misclassify its inputs to a preset-specific label without affecting its original task.
- Attackers poison the model by injecting triggers into the training data that activate the backdoor in the test phase.

Backdoor Attack in Centralized GNNs [5] [6] [7]

- These works focus on GNN models in centralized training.
- The trigger is a subgraph which is defined by
  - Trigger size: the number of nodes in the subgraph
  - Trigger density: the complexity of the subgraph, which ranges from 0 to 1.
  - Node features: the feature vector for each node in the subgraph (optional)

[5] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong (2021). "Backdoor attacks to graph neural networks". In: *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*

[6] Z. Xi, R. Pang, S. Ji, and T. Wang (2021). "Graph backdoor". In: *USENIX Security*

[7] J. Xu, M. Xue, and S. Picek (2021). "Explainability-based backdoor attacks against graph neural networks". In: *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*

Backdoor Attack in Federated Learning

- Many attacks have applied in federated learning on the Euclidean data, e.g., images and words.
  - CIFAR-10, Reddit dataset [8]
  - Fashion-MNIST, UCI Adult Census dataset [9]
  - LOAN, MNIST, CIFAR-10, Tiny-imagenet [10]

[8] "How to backdoor federated learning" (2020). In: *AISTATS*

[9] "Analyzing federated learning through an adversarial lens" (2019). In: *ICML*

[10] "Dba: Distributed backdoor attacks against federated learning" (2019). In: *ICLR*

Challenges of implementing backdoor attacks in Federated GNNs

- The malicious updates will be weakened in the aggregation function.
- The backdoor trigger generation methods and injecting position are different between graph data and images/words.
  - There is no position information we can exploit in the graph data because it's a non-Euclidean data.
- Current defenses may not be effective in backdoor attacks in Federated GNNs.

Backdoor attacks in Federated GNNs

- Therefore, we should expect different behavior of backdoor attacks in Federated GNNs.
- Also, it is crucial to investigate if existing countermeasures that have been tested mostly with Euclidean data are still effective for backdoor attacks in Federated GNNs.

How to design backdoor attacks in Federated GNNs

Definition (Local Trigger & Global Trigger.)

The local trigger is the specific graph trigger for each malicious client in DBA. The global trigger is the combination of all local triggers.[a]

[a] Since it is an NP-hard problem to decompose a graph into subgraphs [11], we first generate local triggers and then compose them to get the global trigger used in CBA.

[11] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani (2008). *Algorithms*. McGraw-Hill Higher Education New York

Two backdoor attacks in Federated GNNs

Definition (Distributed Backdoor Attack (DBA).)

There are multiple malicious clients, and each of them has its local trigger. Each malicious client injects its local trigger into its training dataset. All malicious clients have the same backdoor task. An adversary $\mathcal{A}$ conducts DBA by compromising at least two clients in FL.

Definition (Centralized Backdoor Attack (CBA).)

A global trigger consisting of local triggers is injected into one client's local training dataset. An adversary $\mathcal{A}$ conducts CBA by usually compromising only one client in FL.

Backdoor attack framework

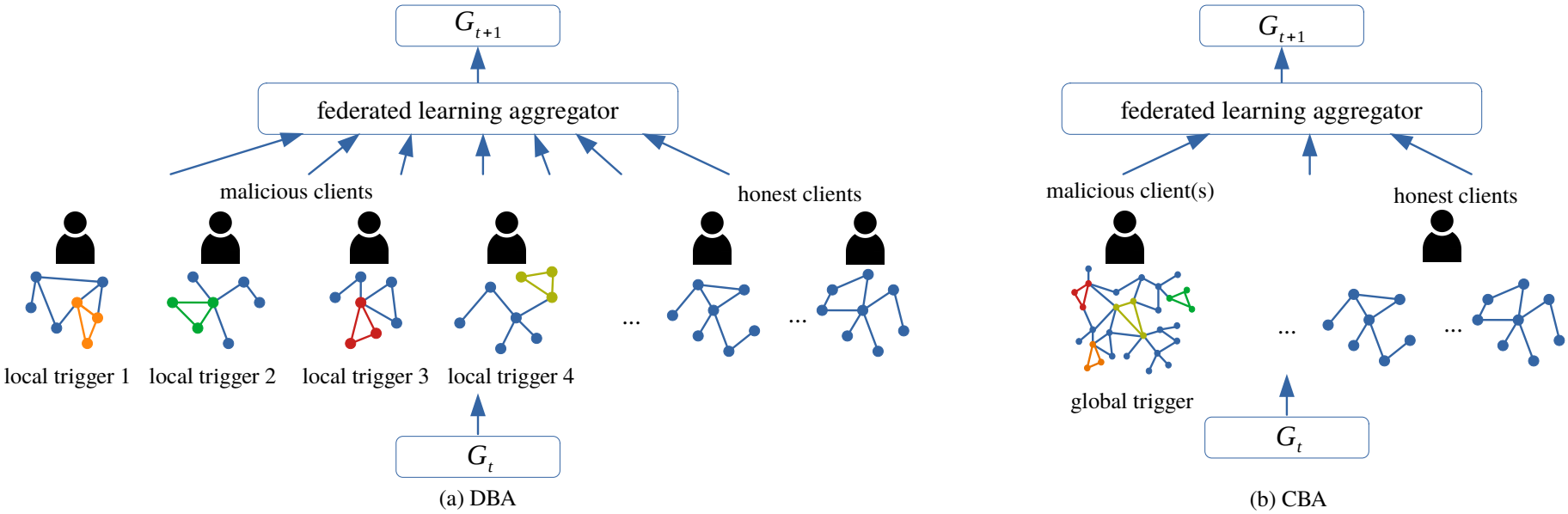


Figure 4: Attack Framework

Trigger generation

- We need to make sure the trigger pattern in CBA is the union set of local trigger patterns in DBA.
  1. First generate local triggers in DBA and then combine them to get the global trigger
  2. First generate a global trigger in CBA and then divide it into M local triggers.
- It is an NP-hard problem to divide a graph into several subgraphs [12]
- Therefore, DBA should be implemented before the CBA.

[12] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani (2008). *Algorithms*. McGraw-Hill Higher Education New York

Trigger generation

- Erdős-Rényi (ER) model $G(n, p)$
- A graph is constructed by connecting nodes randomly.
- The graph has n nodes and each edge is included in the graph with probability p.

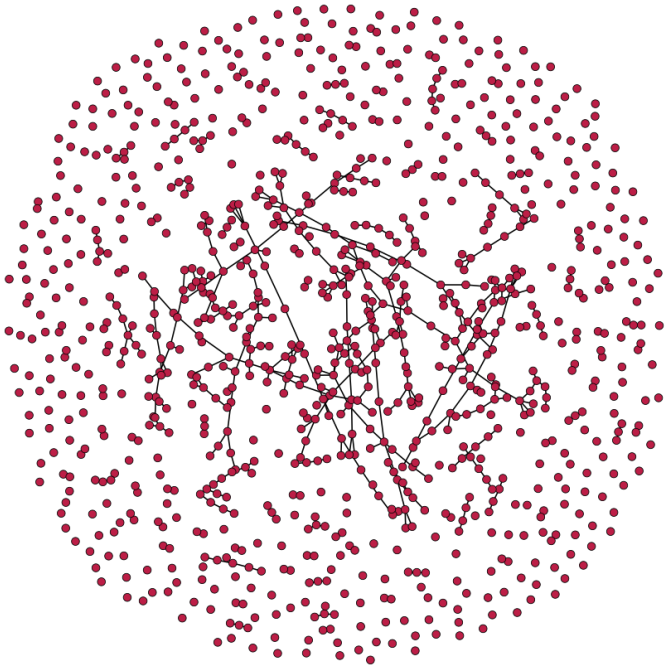


Figure 5: An ER graph with 1000 vertices at the critical edge probability $(1/(n-1))$.

Research Questions

1. What is the behavior of backdoor attacks in Federated GNNs?
   - In CBA, whether the ASR of local triggers can achieve similar performance to the global trigger even if the centralized attacker would embed a global trigger into the model.
   - In DBA, whether the ASR of the global trigger is higher than all local triggers even if the global trigger never actually appears in any local training dataset, as mentioned in [13].
2. What is the impact of different parameters?
   - Number of clients
   - Percentage of malicious clients
   - Attack related parameters: trigger size, poisoning intensity and trigger density
3. How effective are existing countermeasures?

[13] "Dba: Distributed backdoor attacks against federated learning" (2019). In: *ICLR*

Datasets and GNN models

- TUDataset: A collection of benchmark datasets for learning with graphs.
- The datasets can be accessed using the Deep Graph Library which is one of the most popular deep learning libraries for graph data processing.
  - `import dgl.data`
  - `dataset = dgl.data.TUDataset('NCI1')`

Table 1: Datasets statistics.

Dataset	# Graphs	Avg. # nodes	Avg. # edges	Classes	Class Distribution
NCI1	4,110	29.87	32.30	2	2,053[0], 2,057[1]
PROTEINS_full	1,113	39.06	72.82	2	663[0], 450[1]
TRIANGLES	45,000	20.85	32.74	10	4,500[0 – 9]

- GNN models: GCN, GAT, GraphSAGE

Evaluation metrics

- *Attack success rate (ASR)*
  The trigger-embedded inputs are

$$D_{g_t} = \{(G_{1,g_t}, y_1), (G_{2,g_t}, y_2), \ldots, (G_{n,g_t}, y_n)\}.$$

  Formally, ASR is defined as:

$$\textit{Attack Success Rate} = \frac{\sum_{i=1}^{n} \mathbb{I}(G_{backdoor}(G_{i,g_t}) = y_t)}{n},$$

- *Clean accuracy drop (CAD)*: the classification accuracy difference between global models with and without malicious clients over the clean testing dataset.

Experimental artifacts

- Code of all experiments is published in github.
  https://github.com/xujing1994/bkd_fedgnn

Table 2: Summary of the experimental setting (K: number of clients, M: number of malicious clients).[14]

Experiment	Dataset	K	M
Exp. I	NCI1, PROTEINS_full, TRIANGLES	5	2
Exp. II	NCI1, PROTEINS_full, TRIANGLES	5	3
Exp. III	TRIANGLES	10	4, 6
		20	8, 12
Exp. IV	TRIANGLES	100	5, 10, 15, 20

[14] Exp. I, Exp. II, Exp. III, and Exp. IV represent the experiments of honest majority attack scenario, malicious majority attack scenario, the impact of the number of clients, and the impact of percentage of malicious clients, respectively.

Required software packages

- `torch>=1.9.0`
- `torchvision>=0.10.0`
- `numpy>=1.23.2`
- `dgl>=0.9.1`
- `networkx>=2.4`
- `hdbscan==0.8.28`
- `joblib==1.1.0`

How hard was it to reproduce the original results?

- The github repository documentation is well detailed and easy to follow.
- All scripts are runnable and cover all components relevant to the experimental section of the paper.
- The obtained results are reproducible.
- The data used in the experiments are already processed. There is no guide on the data pre-processing scripts so that users can easily adapt their own datasets to the provided algorithms.

Script examples

- Train a clean Federated GNN model
  `python clean_fedgnn.py --dataset NCI1 --config ./configs/TUS/TUs_graph_classification_GCN_NCI1_100k.json --num_workers 5 --num_mali 0 --filename ./Results/Clean`
- The results will be saved in the folder ./Results/Clean which will be used to calculate the clean accuracy drop.

Script examples

- Implement distributed backdoor attack in Federated GNNs

```
python dis_bkd_fedgnn.py --dataset NCI1 --config
./configs/TUS/TUs_graph_classification_GCN_NCI1_100k.json
--num_workers 5 --num_mali 2 --filename ./Results/DBA
```

- Implement centralized backdoor attack in Federated GNNs

```
python cen_bkd_fedgnn.py --dataset NCI1 --config
./configs/TUS/TUs_graph_classification_GCN_NCI1_100k.json
--num_workers 5 --num_mali 2 --filename ./Results/CBA
```

Script examples

Output of the centralized backdoor attack in Federated GNNs (one epoch):

```
epoch: 0
Client 0, loss 0.6251, train acc 0.628, test loss 0.6968, test acc 0.492
Client 0 with global trigger: 1.000
Client 0 with local trigger 0: 1.000
Client 0 with local trigger 1: 1.000
Client 1, loss 0.5336, train acc 0.714, test loss 0.6824, test acc 0.562
Client 1 with global trigger: 1.000
Client 1 with local trigger 0: 1.000
Client 1 with local trigger 1: 1.000
Client 2, loss 0.8846, train acc 0.489, test loss 0.6891, test acc 0.585
Client 2 with global trigger: 1.000
Client 2 with local trigger 0: 1.000
Client 2 with local trigger 1: 0.901
Client 3, loss 0.6916, train acc 0.543, test loss 0.6999, test acc 0.467
Client 3 with global trigger: 0.758
Client 3 with local trigger 0: 0.714
Client 3 with local trigger 1: 0.033
Client 4, loss 0.6709, train acc 0.604, test loss 0.7165, test acc 0.467
Client 4 with global trigger: 0.099
Client 4 with local trigger 0: 0.110
Client 4 with local trigger 1: 0.011
Global Test Acc: 0.579
Global model with global trigger: 1.000
Global model with local trigger 0: 1.000
Global model with local trigger 1: 0.736
```

Figure 6: Output of CBA in Federated GNNs.

Results

`GCN_NCI1_5_2_0.20_0.20_0.80_global_attack.txt`

Table 3: Saved results of DBA on NCI1 datast (5 clients, 2 malicious clients).

ASR with global trigger	ASR with local trigger 1	ASR with local trigger 2
0.777	0.798	0.894
0.734	0.766	0.862
0.840	0.872	0.926

Plot figures

- For each experimental setting, we repeated the experiment 10 times to eliminate randomness introduced from our algorithms.
- We use `numpy.average` and `numpy.std` to calculate the average value and standard deviation.
- `matplotlib.plot` and `matplotlib.fill_between` to plot the figures. For example:
  - $ax.plot(x, y)$
  - $ax.fill\_between(x, y - std, y + std)$

Running environment

- PyTorch (1.9.0+cu111)
- We run our experiments in TU Delft's cluster consisting from CentOS-based worker machines.

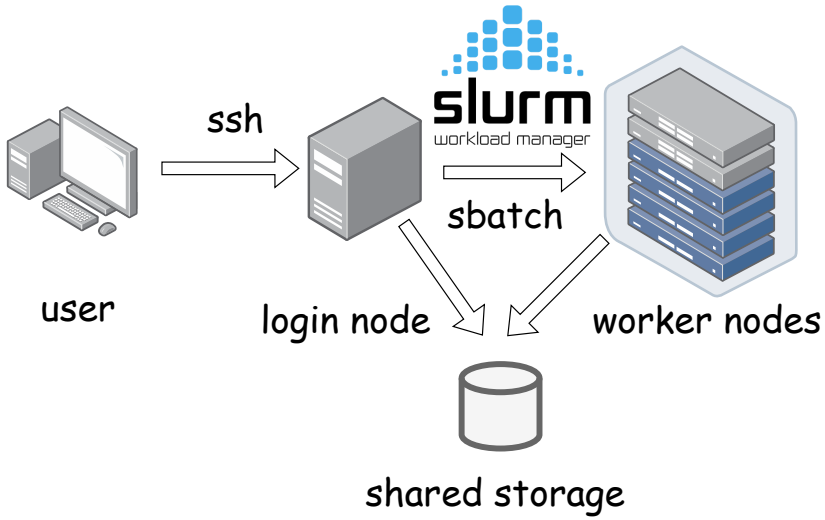


Figure 7: Cluster's architecture

- There are multiple NVIDIA GPUs (Tesla P100 and V100, GeForce GTX 1080 Ti and GTX 2080 Ti, A40).
- Our code can be run on Google Colab without any issues.

Challenges regarding the environment

- Running experiments in ML is resource intensive and takes time.
- Our cluster is shared so we need to make sure that everything is correct before we run the experiments there.

Methodology

To eliminate our implementation mistakes we followed the following steps:

- Training:
  1. From small datasets to large.
  2. From centralized to distributed.
- Backdoor:
  1. First run experiments with extremes (large poisoning rate and large trigger size) to make sure that the poisoning pipeline works.
  2. Move to distributed setup.
  3. Use more realistic poisoning hyperparameters.

Findings

CBA:

- In the global model the ASR of all local triggers can be as high as the global trigger, which is counterintuitive as the centralized attack only embeds the global trigger into the model.
- In the malicious local model, the ASR of all local triggers is already close to the global trigger's ASR, meaning that the malicious local model has learned the pattern of each local trigger.

DBA:

- The attack success rate of the global trigger is higher than (or at least similar to) any local trigger, even if the global trigger never actually appears in any local training dataset.

Failure case

- GCN (mean aggregation function) + TRIANGLES → bad classification performance
- Solution: Change the `Aggregation` function from default `mean` to `sum`
- Reason: Due to the property of the TRIANGLES dataset, the aggregation function of `sum` works better than `mean`

What did we learn

- Comment your code and use descriptive commit messages because you can easily forget the reason behind some of your decisions.
- We learned the hard way that we need to document everything that we run to avoid wasting time in the same experiments.

Discussion

- We are the first to implement a backdoor in federated GNNs so we did not have a baseline.
  - Do you create your own baseline in such cases?
- Security of ML is most of the time empirical. How can we ensure the current experiments are enough for a paper?
- How do you do comparisons? What metrics do you use?
- How do you run ML experiments? Do you also use slurm workload manager? What can we do when the resources are shared and limited sometimes?

Impact of number of clients

- For the largest dataset (TRIANGLES), we set bigger number of clients, i.e., 10 and 20.
- With more clients, the attack success rate of CBA decreases while the attack performance of DBA keeps steady.

Impact of percentage of malicious clients

- More clients and less percentage of malicious clients on the large dataset - TRIANGLES, e.g., 100 clients and fewer malicious clients, ranging from 5% to 20%.
- Pearson Correlation Coefficient (PCC)
  - Compute the ASR under different number of malicious clients
  - Apply `numpy.corrcoef(NM, ASR)` where NM: number of malicious clients.
  - The increase in NM has more positive impact in DBA than CBA.

Impact of backdoor hyperparameters

- Effects of trigger size: not the number of nodes in a trigger graph, but γ fraction of the graph dataset's average number of nodes.
- Effects of poisoning intensity: the ratio that controls the percentage of backdoored training dataset among the local training dataset.
- Effects of trigger density: the complexity of a local graph trigger, which ranges from 0 to 1.

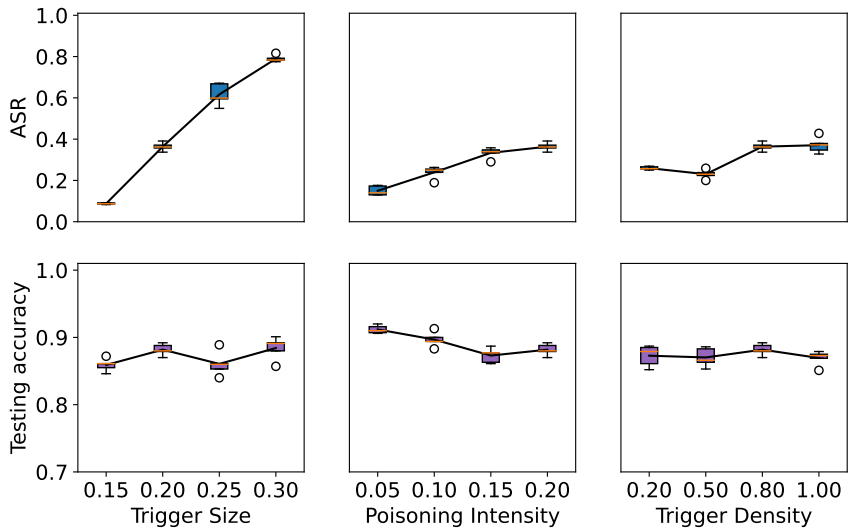


Figure 8: Results on TRIANGLES with different trigger parameters.

How effective are existing countermeasures?

- FLAME: one of the state-of-the-art defenses against backdoor attacks in FL.
  - Filter out outliers, clip high magnitude updates, and add noise to limit the backdoor's effect.
  - No source code available → Re-implementation.
- FoolsGold:
  - Based on the assumption that honest clients can be separated from sybils by the diversity of their gradient updates.
  - One of the assumptions is that each client's training data is non-i.i.d and has a unique distribution → fits the non-i.i.d data distribution setting in our work.

How effective are existing countermeasures?

- Both defenses are not very effective against our attack.
- Cosine similarity is not very suitable for graph data.
- Under FoolsGold, there is a significant increase in CBA's ASR in all models, but the testing accuracy of CBA reduces significantly at the same time.

Hypothesis

Under FoolsGold, the malicious client in CBA is assigned a higher weight. (The idea of FoolsGold: it reduces the aggregation weights of detected malicious clients while retaining the weights of other clients.)

Table 4: FoolsGold weight in DBA and CBA on TRIANGLES.

Attacks	Attacker 1	Attacker 2 (client 2 in CBA)	Client 3	Client 4	Client 5	Attackers (sum)
DBA	0.57 ± 0.23	0.57 ± 0.23	0.86 ± 0.13	0.86 ± 0.13	1.00 ± 0.00	1.14 ± 0.23
CBA	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00

What did we learn

- Re-implementation is oftentimes required
  - Represents extra effort
  - May fail to respect original implementation decisions (some tricks may needed to get the original results).
  - Make your code available and provide documentation for reproduction.
- The experimental results are not always as expected.
- Additional experiments are required to explain the current experimental results.
- The methods which are proved to be effective in one domain may not work well in other domains.
- In ML it is very useful to save the models after training to avoid spending time again when we evaluate them against defenses or run analysis experiments with them.

Discussion

- How to choose the defences/attacks to evaluate the robustness of your proposed method?
  - The latest ones?
  - The most effective ones?
  - How to reproduce the methods? Are they open source? How to guarantee the same experimental environment and setting?
- Generalization vs Specialization.
- Do you save the trained models from your experiments? What about the space requirements? Do you have any efficient ways to save large models?

Going Forward

- Look towards future developments on large, realistic datasets
  - What if you could experiment with your own graph datasets?
- Add a data processing script to help community when using custom datasets.
- Test Federated GNNs against more defences.

Contact

Jing Xu	j.xu-8@tudelft.nl
Stefanos Koffas	s.koffas@tudelft.nl
Stjepan Picek	s.picek@tudelft.nl