THE PRICE OF FORGED EVIDENCE IN SIMILARITY-BASED DEFENSES FOR GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Similarity-based defenses for graph neural networks decide whether an edge is genuine by comparing the features of its endpoints. If the attacker can also write those features, that evidence can be forged. Adaptive attacks are known to break such defenses; what has not been measured is how cheap the forgery is and how much of the damage it explains. We price it exactly: for Jaccard and cosine filters we give the fewest feature changes that make an inserted edge pass, in closed form, and **about one copied word for each inserted edge the filter would remove makes GCN-Jaccard keep every edge that Metattack or PRBCD inserts**, with no change to the attack, for at most 0.46% of the non-zero feature entries on the citation graphs (1.31 to 5.84% on the heterophilous ones). The defense then does harm: on Cora, 136 copied words added to Metattack's graph leave GCN-Jaccard losing more accuracy than an undefended GCN (15.7 against 13.3 points). We also measure forgery edge by edge: we count which inserted edges each defense keeps, and rerun the defense on clean features while its model still trains on poisoned ones, which separates *blinding* from ordinary feature damage that no edge filter was built to stop; the pruning counts this literature reports conflate the two. Every countermeasure we test that reads raw features, a higher threshold, a rule that discounts shared words, an edge-aware corroboration check, and GNNGuard's stricter cosine, only raises the price, while the defender pays in clean edges. Defenses that learn their evidence first are the exception: on Cora and CiteSeer the same copies change neither a STABLE-style purifier nor SimP-GCN by more than 2.8 points, though a large feature budget still blinds them. A joint bilevel attack, CLASP, finds the forgery without seeing the defense. The pattern holds across six graphs spanning homophily 0.22 to 0.81, wherever the attacker's edges do damage.

1 INTRODUCTION

Graph neural networks derive predictive power from relational structure, and that dependency is a training-time attack surface: an adversary who perturbs a small fraction of edges can alter message passing, model selection, and generalization (Zügner et al., 2018; Zügner and Günnemann, 2019; Geisler et al., 2021). Global poisoning attacks have largely asked *which graph elements to perturb* under a structural budget. Robust preprocessing defenses, in turn, typically assume node attributes are trustworthy side information for judging whether an observed edge is plausible. We study what happens when that independence assumption is removed. If a defense decides an edge is suspicious by comparing endpoint features, and the attacker may poison those same features before training, then the defense's similarity signal is no longer independent evidence. It is part of the attack surface. We ask whether an adversary that jointly poisons structure and node attributes can, besides degrading a GNN, manipulate the very evidence a similarity-based defense uses to detect the attack.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

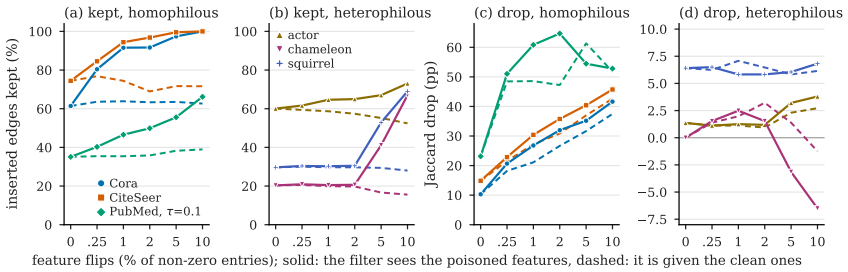

Figure 1: Joint poisoning blinds GCN-Jaccard on all six graphs (Sec. 5.2; values in Table 8). Solid is the filter as deployed, dashed the same filter given the clean features while its GCN still trains on the poisoned ones. (a,b) The kept share of CLASP's inserted edges, with PubMed at $\tau = 0.1$ where the filter is active. (c,d) Accuracy drop: the gap between the curves is blinding, the rise of the dashed curve is feature damage. A negative drop in (d) is the defended model beating its own clean accuracy, where the filter removes most of a heterophilous graph's edges.

We answer yes. On six graphs we measure which of the attacker's edges each defense keeps, instead of inferring failure from accuracy alone, and we test three rules the mechanism suggests as remedies, each against an attacker told about it.

Contributions.

- **A measurement of forgery, edge by edge.** Rather than inferring a defense's failure from accuracy, we count which of the attacker's inserted edges each defense keeps, and we run every defense twice on the same poisoned graph: once on the features the attacker wrote and once on the clean ones, while the victim always trains on the poisoned ones. The gap is what corrupting the evidence buys; the rest of the loss is feature damage no edge filter was designed to stop. On Cora that gap is at most 5.7 points for GCN-Jaccard and 9.7 for GNNGuard, and the pruning counts this literature reports conflate the two (Sec. 5.2).
- **The price of forgery in closed form, and what every countermeasure does to it.** Proposition 1 gives the fewest feature flips that lift an edge past a Jaccard threshold, exactly and tightly; Proposition 2 gives the smallest ℓ_2 move past a cosine one. One copied word per inserted edge suffices on Cora at Jaccard's default threshold, and such copies make GCN-Jaccard keep every edge Metattack or PRBCD inserts. Every countermeasure we test that reads raw features answers with a price rise and nothing more: a higher threshold, a rule that discounts an edge's k most helpful shared words, GNNGuard's stricter cosine, and an edge-aware corroboration check, which catches the minimal copies (AUC 0.835 on Cora, and pruning on it recovers everything they cost) until about one and a half further words per edge, after which the tail the attack occupies inverts and the sign of a one-sided test is the attacker's to choose (Sec. 5.3).
- **Where blinding stops, and what remains.** Learning the evidence first is the one defense family the cheap copies do not buy: on Cora and CiteSeer they change neither a STABLE-style purifier nor SimP-GCN by more than 2.8 points (App. H), and an attacker handed the purifier and aiming at its embedding buys only 1.67 to 2.47 points (App. I). What does blind them is a feature budget large enough to change what the nodes say, which **CLASP**, a joint bilevel attack that never sees the defense, supplies: at a 10% budget GCN-Jaccard keeps all of its edges on Cora and CiteSeer, GNNGuard goes from 15% to 89% on Cora, the learned purifier from 62% to 85%, and SimP-GCN ends below an undefended GCN on three graphs. The cost of blinding follows the damage those edges do (Secs. 5.2 and 5.4).

What is new. That similarity defenses can be beaten is not new: attacks adapted to a defense strip many of them of their apparent robustness (Mujkanovic et al., 2022), graph injection passes homophily-based defenses once it preserves homophily (Chen et al., 2022), and Nettack and Wu et al. (2019) already perturb edges and features together. What is new is the mechanism and its price: the evidence a raw-feature filter reads can be forged after the fact, on top of any structural attack, without changing it or touching the victim's training, at a price Proposition 1 gives exactly per edge; and kept edges, measured with and without the forged evidence, split a defended model's

loss into blinding and feature damage, which accuracy and pruning counts cannot. CLASP is the tool, not the claim: it reaches feature budgets the published structural attacks lack, and App. Q reports where it loses, including that it is far weaker than PRBCD on every heterophilous graph.

2 Related Work

Graph poisoning and bilevel optimization. Nettack introduced discrete adversarial perturbations for graph data (Zügner et al., 2018); Metattack formulated global poisoning as bilevel optimization differentiated through surrogate training (Zügner and Günnemann, 2019). PGD and MinMax optimize a continuous relaxation of edge flips by projected gradient descent (Xu et al., 2019); PRBCD and GRBCD improved scalability by optimizing over a randomly resampled block of candidate entries rather than all n^2 pairs (Geisler et al., 2021); Meta-PGD runs projected gradient descent on meta-gradients (Mujkanovic et al., 2022), and LR-BCD adds node-level budgets to PRBCD (Gosch et al., 2023a); GOttack, a targeted attack, restricts candidates by graphlet-orbit signature (Alom et al., 2025); EvA replaces gradients with evolutionary search (Akhondzadeh et al., 2026). Metacon debiases the meta-gradient surrogate objective with a contrastive term (Yoon et al., 2024); it is the recent meta-gradient attack closest to ours with public code, so App. F runs it in its own harness.

CLASP keeps Metattack's bilevel backbone but samples a class-conditioned candidate pool and exposes structure and features to one outer objective; Sec. 5.4 swaps in PRBCD's uniform sampling.

Adaptive attacks, unnoticeability, and prior joint attacks. Mujkanovic et al. (2022) showed that many graph defenses lose their apparent robustness against attacks adapted to them, and Chen et al. (2022) that homophily-based defenses stop graph injection until the attacker adds a homophily-preserving objective (HAO). Both adapt the attack to the defense. CLASP never sees the defense; the camouflage of Sec. 5.3 is deliberately defense-aware, and its point is the price. Wu et al. (2019), who introduced the Jaccard filter, attacked edges and features with integrated gradients in a targeted setting and found feature perturbations much weaker; Nettack also perturbs both (Zügner et al., 2018), and bilevel poisoning has been extended to self-supervised embeddings (You et al., 2026). Poisoning perturbations concentrate around training nodes (Li et al., 2023), and common budgets can change a graph's semantics (Gosch et al., 2023b); Secs. 5.2 and 5.3 measure both for our flips.

Defenses. GCN-Jaccard removes edges with low endpoint-feature similarity (Wu et al., 2019), GCN-SVD performs low-rank preprocessing (Entezari et al., 2020), RGCN models hidden representations probabilistically (Zhu et al., 2019), Pro-GNN jointly learns a cleaned adjacency and a GNN (Jin et al., 2020), GNNGuard reweights and prunes edges by the cosine similarity of endpoint representations at every layer (Zhang and Zitnik, 2020), SimP-GCN preserves feature similarity inside the network (Jin et al., 2021), and recent work learns a self-supervised purifier (Lee and Park, 2025). We ask whether a purification rule stays informative when the attacker can modify the evidence it consumes.

3 CLASP

CLASP (**CLA**ss-conditioned **S**tructure-feature **P**oisoning) conditions its structural search on pseudo-labels and clasps structure and features to one outer objective, with one unrolled surrogate producing both gradients.

3.1 Threat model

Let $G = (A, X)$ be an attributed undirected graph with $A \in \{0,1\}^{n\times n}$ and binary features $X \in \{0,1\}^{n\times d}$. The attacker observes the training graph and the labeled training nodes, poisons the data before victim training, and does not require the victim architecture. The structural budget is

$$B_E = \lfloor 0.05\,|E| \rfloor, \tag{1}$$

where $|E|$ counts undirected edges. Feature budgets are counted as binary entry flips and reported *relative to the number of non-zero feature entries* (X_{nz}, the unit used in every table), not the dense $n \times d$ matrix (on Cora, 49,216 of 3,880,564 entries). Structure-only rows set the feature budget to

exactly zero ($B_X = 0$), so comparisons with Metattack, PGD, MinMax, PRBCD and GRBCD are like-for-like; rows labelled *joint* always state their feature budget. Like Metattack's singleton filter, we forbid removals that would isolate a node; since a block flips several edges at once, a repair step restores one incident edge for any node left isolated (App. Q audits this).

3.2 Class-conditioned structural search

A clean surrogate first produces pseudo-labels $\hat{y}$ (train labels on labeled nodes, surrogate predictions elsewhere, following the self-training protocol of Metattack). For each pseudo-class c we sample pairs (u, v) with $\hat{y}_u = c$, $\hat{y}_v \neq c$, and merge them with all existing undirected edges so that removal remains reachable. Duplicates are removed. On heterophilous graphs, with edge homophily $h \leq 0.5$, the rule switches to same-class pairs, $\hat{y}_v = c$. The name records this rule, not a claimed advantage: Sec. 5.4 finds it helps on one graph of six. Our runs set this switch from h computed with all labels, one bit beyond the attacker's view; h computed from the pseudo-labels makes the same choice on all six graphs, and rerunning Cora and CiteSeer with it reproduces the stored graphs edge for edge.

Because the pool grows with n rather than n^2, no dense $n \times n$ gradient is ever formed (App. Q).

3.3 Joint bilevel co-poisoning objective

We use a two-layer linearized GCN surrogate $\hat{A}^2 XW$ and unroll $T = 50$ inner optimization steps with momentum. Let $z_i \in [0, 1]$ be the relaxed state of candidate edge i. After unrolling, the outer loss is evaluated on unlabeled nodes using pseudo-labels. The direction-corrected structural score is

$$s_i = \mathrm{zscore}\left(\frac{\partial \mathcal{L}_{\text{out}}}{\partial z_i} (1 - 2z_i) \right). \tag{2}$$

The highest-scoring candidates are flipped in five equal blocks, and the remainder of the budget after division by five is left unspent, so 260 of Cora's 263 budgeted edges change in practice. The *same* outer objective is differentiated with respect to binary node attributes, giving

$$r_{uj} = \frac{\partial \mathcal{L}_{\text{out}}}{\partial X_{uj}} (1 - 2X_{uj}), \tag{3}$$

and the highest-scoring previously unflipped entries are toggled under budget B_X. Both scores come from one backward pass through the same unrolled surrogate, so each block changes connectivity and the attributes a similarity defense later consults together.

4 Experimental Setup

Datasets and splits. We evaluate on six graphs (Table 25): the homophilous citation networks Cora, CiteSeer (Sen et al., 2008) and PubMed with the public Planetoid split (Yang et al., 2016), and the heterophilous chameleon, squirrel (Rozemberczki et al., 2021) and actor with the first geom-GCN split (Pei et al., 2020). The two protocols differ in more than homophily: the public split labels 3.6% of CiteSeer's nodes, 5.2% of Cora's and 0.3% of PubMed's, the geom-GCN split 48%. Much of the Metattack and Pro-GNN literature instead uses a random $10/10/80$ split on the largest connected component, so every baseline is rerun in our setting, and App. F runs Metacon inside that other protocol. Feature budgets are counted as a fraction of non-zero feature entries (Sec. 3). PubMed's TF-IDF features are binarized to word presence wherever the Jaccard filter or the feature channel is involved, for attack and victim alike.

Victims and defenses. We evaluate a plain GCN (Kipf and Welling, 2017); GCN and GAT (Veličković et al., 2018) with a rank-8 residual low-rank adapter after each layer, in the parameterization of LoRA (Hu et al., 2022) but trained from scratch together with the base weights, as two architectures that differ from the attack's linear surrogate (*GCN+Ad.*, *GAT+Ad.*); the DeepRobust (Li et al., 2020) implementations of GCN-Jaccard, GCN-SVD, RGCN and Pro-GNN; SimP-GCN; and GNNGuard, run from its authors' code with the settings of their Metattack experiments. Mechanism analysis centres on GCN-Jaccard because its rule is an explicit function of endpoint feature similarity.

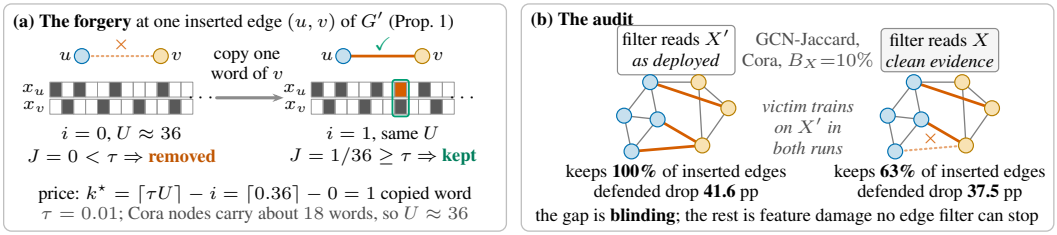

Figure 2: The forgery and the measurement. (a) GCN-Jaccard keeps an edge only if its endpoints' word sets overlap by τ; two Cora nodes with no shared word have $U \approx 36$, so one copied word suffices (Prop. 1). (b) The filter reads the poisoned or the clean features while its GCN trains on the poisoned ones; the gap is blinding (Cora, 10% budget, Table 1).

Evaluation and cost. Each poisoned graph is scored by retraining victims with three, five or ten seeds (each caption states its count), taking test accuracy at the best validation epoch; these seeds measure victim-training variability only, and Table 7 regenerates CLASP and PRBCD on all six graphs, and Metattack on Cora, with independent attack seeds. Attack time includes training any surrogate, and peak memory is measured in a separate process per attack, on one 2-core CPU machine.

5 Results

5.1 Where CLASP stands, and what it costs

CLASP's structural component and PRBCD are the two strongest structure-only attacks on Cora and CiteSeer, level on Cora over ten attack seeds and ahead on CiteSeer, while on all three heterophilous graphs the ordering reverses and CLASP is far weaker than PRBCD; it is the fastest optimization-based attack on Cora and the most memory-hungry on five of six graphs (App. Q). Adding the feature channel takes Cora and CiteSeer to 41.06 and 47.80 pp at a 10% feature budget, and GCN-Jaccard, which helps against structure alone (10.46 against 17.56 pp on Cora), gives nothing against it (41.62; Table 4). The rest of this section measures why, and none of it depends on CLASP being the strongest attack.

5.2 Joint poisoning blinds the filter, edge by edge

GCN-Jaccard keeps an edge only if the Jaccard similarity of its endpoints' feature sets reaches τ; at DeepRobust's default $\tau = 0.01$ and about 18 words per Cora node, that means sharing at least one word. We re-scored every stored poisoned graph edge by edge: Table 1 covers all six graphs at feature budgets of 0 and 10%, and Figure 1 and Table 8 every budget on every graph.

The filter keeps the attacker's edges. On every graph, flipping 10% of the feature entries makes the filter keep more of CLASP's inserted edges, 66 to 100% of them, while given the clean features it would keep 16 to 72% (Table 1; on PubMed at the threshold where the filter is active, below). On Cora, where the filter removes 39% of the inserted edges against structure alone, flipping 0.25% of the feature entries already raises the share kept from 61% to 80%, and at 10% it keeps every one, although given the clean features it would still remove about a third of the same edges; CiteSeer goes from 74% to 85% and then to all of them. Three further attack seeds leave this unchanged (Table 23): over four draws the filter keeps $64 \pm 5\%$ and $73 \pm 5\%$ against structure alone and $100 \pm 0\%$ at a 10% budget, against 63 ± 0 and 69 ± 4 given the clean features. The onset differs by graph: 0.25% also lifts the share on PubMed, from 35 to 40%, while on chameleon and squirrel it stays where structure alone put it until 2% and rises only from 5% (Table 8). The attack never sees the defense, yet at 0.25%, 70% of its flips land on endpoints of its own inserted edges, which are 12% of nodes, and 65% of the words it adds are words the inserted neighbour already has (59% and 43% on CiteSeer). The surrogate $\hat{A}^2XW$ is a plausible reason: once two nodes are joined, the feature gradient flows through the new edge and pushes both endpoints toward the same words. Two thirds of these first flips land on labelled nodes, where poisoning is known to concentrate (Li et al., 2023); protecting those nodes removes most of the effect (App. B).

Table 1: Joint poisoning and GCN-Jaccard on all six graphs: CLASP's graphs at a 5% edge budget with no feature flips (0) and with 10% of the non-zero feature entries flipped (10), three victim seeds. *Kept*: share of the inserted edges the filter keeps, given the poisoned features and, at 10, given the clean ones. Drops are in points against each model's clean accuracy; *clean ev.* gives the filter the clean features while its GCN trains on the poisoned ones. The filter uses DeepRobust's default $\tau = 0.01$, which is nearly inert on PubMed, so PubMed also appears at $\tau = 0.1$. Table 8 gives every budget and Table 11 the same measurements under the geom-GCN split.

	inserted edges kept			undefended		GCN-Jaccard drop (pp)		
Graph	0	10	10, clean ev.	0	10	0	10	10, clean ev.
Cora	61%	100%	63%	17.6	40.9	10.3	41.6	37.5
CiteSeer	74%	100%	72%	19.1	46.5	14.9	45.7	42.6
PubMed	99%	100%	99%	55.9	55.3	51.8	52.7	52.8
PubMed, $\tau = 0.1$	35%	66%	39%	55.9	55.3	23.1	52.8	51.8
actor	60%	73%	52%	0.7	2.5	1.4	3.8	2.7
chameleon	20%	67%	16%	7.7	5.7	0.0	−6.5	−1.2
squirrel	30%	69%	28%	16.5	17.4	6.4	6.8	6.1

Blinding and damage are different, and the pruning count conflates them. Giving the filter the clean features while its GCN still trains on the poisoned ones (last column of Table 1; Fig. 2b) isolates what the corrupted evidence costs: it is real but bounded, 2.5 points at 0.25% and at most 5.7 at any budget on Cora (about 5 on CiteSeer), and the rest of the defended model's loss comes from the poisoned features, which no edge filter can stop. The count of removed edges mixes the two: on Cora it falls from 670 to 432 between the structure-only and 10% joint graphs, but only 98 of that fall are inserted edges it now keeps.

GNNGuard is blinded the same way. GNNGuard weights every edge by the cosine similarity of its endpoints and drops those below 0.1, at its first layer from the input features. Run from its authors' code on the same graphs (Table 12), it is a far stricter filter, keeping only 15% of CLASP's structural edges on Cora and 10% on CiteSeer. Joint poisoning undoes that: at a 10% feature budget its first layer keeps 89% and 87% of the inserted edges, against 18% and 10% given the clean features, and blinding costs it more than it costs Jaccard: computing only the first layer's weights from the clean features recovers 9.7 of GNNGuard's 40.3-point loss on Cora and 4.8 of 41.1 on CiteSeer. The heterophilous graphs blind it the same way and damage it differently: on chameleon its first layer keeps 3% of CLASP's structural edges and 39% at a 10% budget, while the cost comes from the cosine copies, which take PRBCD's edges from 27 of 1,304 kept to all of them and GNNGuard's loss from 0.4 to 30.7 pp (on squirrel, 4.2 to 32.7).

PubMed. The default threshold is nearly inert on PubMed: its nodes carry about 50 words, so almost any two share one, and the filter keeps 99% of CLASP's inserted edges before any flip. At $\tau = 0.1$, which prunes 14% of clean edges for under a point of clean accuracy, it keeps only 35% of them, and flipping 10% of feature entries raises that share to 66%, against 39% given the clean features (Table 13). PubMed's accuracy drops are not monotone in the feature budget, at this threshold or without the filter: every gradient attack there takes the GCN below chance, so those drops measure how far past the collapse a run lands, not how much damage it does.

5.3 A FEW COPIED WORDS BLIND THE FILTER, WHOEVER INSERTED THE EDGES

Proposition 1. *Let an edge join binary feature vectors with i shared words and U words in their union, and let a filter keep it iff $i/U \geq \tau$. If $i < \tau U$, the fewest feature flips at its two endpoints that make it pass is exactly $k^\star = \lceil \tau U \rceil - i$, and copying words the other endpoint already has is an optimal way to spend them.*

The proof is four lines of case analysis over what one flip can do to i and U (App. S).

The same price exists in closed form for the other filter this paper measures.

Proposition 2 (Price of passing a cosine filter)**.** *Let $x_u, x_v \in \mathbb{R}^d_{\geq 0}$ be non-zero, $\theta = \angle(x_u, x_v)$, and let a filter keep the edge iff $\cos\theta \geq \tau$, with $0 < \tau < 1$. If $\theta > \arccos\tau$, the smallest ℓ_2 change to x_u*

Table 2: The same failure without CLASP, on all six graphs. A structural attack's stored graph receives the fewest copied words that lift every inserted edge to the filter's Jaccard threshold (Prop. 1); no other feature changes. Drops are in points against each model's clean accuracy, three victim seeds. The threshold is 0.01, except on PubMed, where the filter is active only at 0.1. Metattack's graph is camouflaged where Metattack ran (Cora, CiteSeer, chameleon). Table 9 adds CLASP's structural graphs and PubMed at $\tau = 0.01$.

				inserted edges kept		Jaccard drop (pp)	
Graph	Attack	Copies	% X_{nz}	before	after	before	after
Cora	Metattack	136	0.28	47%	100%	7.6	15.7
	PRBCD	86	0.17	67%	100%	10.9	14.9
CiteSeer	Metattack	57	0.05	74%	100%	12.0	13.2
	PRBCD	55	0.05	76%	100%	13.8	16.8
PubMed, $\tau = 0.1$	PRBCD	4,566	0.46	42%	100%	30.0	55.3
actor	PRBCD	536	1.31	57%	100%	8.9	15.6
chameleon	Metattack	1,248	4.28	15%	100%	3.7	13.9
	PRBCD	1,093	3.75	11%	100%	7.1	36.2
squirrel	PRBCD	5,460	5.84	34%	100%	27.3	40.8

alone that makes the edge pass is

$$\delta^\star = \|x_u\|_2 \, \sin\big(\theta - \arccos\tau\big),$$

and the minimiser is non-negative. In particular, if the endpoints share no words ($\theta = \pi/2$), then $\delta^\star = \tau\|x_u\|_2$: the price is linear in τ, as in Proposition 1.

The acceptance set is a convex circular cone about x_v, so this is the distance to a cone (App. S). It prices the continuous case the way Prop. 1 prices the binary one, and App. M evaluates it on PubMed, where the median rejected edge costs 0.4% of one endpoint's norm at $\tau = 0.01$ and the construction used there turns out to overpay by up to 16.7 times.

Two typical Cora nodes with no shared word have $U \approx 36$, and no Cora node has more than 30 words, so at $\tau = 0.01$ one copied word suffices for any edge (Fig. 2a). Table 2 applies this to attacks that never touch features: to each structural attack's stored graph we add the fewest copied words that lift every inserted edge to the threshold, and change nothing else. On Cora, 136 copies (0.28% of non-zero entries) make GCN-Jaccard keep all of Metattack's inserted edges instead of 47%, and its loss rises from 7.6 to 15.7 pp, more than the 13.3 an undefended GCN loses on the same graph. The copies do no damage of their own: the undefended GCN is unaffected, a filter given the clean features loses 7.8 pp, and a graph-free MLP trained on clean features changes its prediction on only 5% of the copied-to test nodes. PRBCD's and CLASP's structural graphs behave the same way (Table 9); on CiteSeer 55 to 57 copies (0.05%) suffice; on the heterophilous graphs fewer than one copy per inserted edge makes the filter keep all of PRBCD's, raising the defended loss on each; and on PubMed at $\tau = 0.1$, 4,566 copies (0.46%) take it from keeping 42% to all of them and its loss from 30.0 to 55.3 pp. GNNGuard's stricter threshold only raises the price: 421 copies on Cora (0.86%) and 500 on CiteSeer (0.48%) make its first layer keep all 263 of Metattack's inserted edges and 226 of 227, and its loss rises from 4.1 to 12.7 pp and from 2.2 to 10.6 (Table 12).

Raising τ does not rescue the filter (Table 15). At $\tau = 0.05$ it already prunes 1,605 clean edges, 30% of Cora's, and gives up 5.5 points of clean accuracy, while 420 copies (0.85%) restore full survival of Metattack's edges: the attacker's cost grows linearly in τ (Proposition 1), the defender's in clean edges. The same trade runs across the other graphs, and for the attacker it is not always cheap: on chameleon 2,522 copies (8.6% of non-zero entries) still get 1,302 of PRBCD's 1,304 inserted edges past $\tau = 0.05$, where the filter already removes 30,432 of the graph's 31,371 edges; on squirrel the same construction costs 22,160 copies, 24% of the non-zero entries. A cosine filter behaves almost exactly like Jaccard at $\tau = 0.01$, since for all but the longest documents both reduce to sharing one word, and does no better against the joint attack at higher thresholds (Table 16). The failure mode is general and cheap; what CLASP adds is that its gradient finds it unprompted, together with feature damage that no edge filter can stop.

The cheapest copies leave a signature, and more words remove it. The copies of Table 2 are minimal, and minimal makes them visible. An edge-aware check scores each surviving edge by

how few words its endpoints share, how far those words are corroborated by each endpoint's *other* neighbours, and how rare they are, each read in the direction the data indicates since an AUC below chance separates as well as the same distance above it. On Cora the strongest reaches 0.835 against Metattack's camouflaged edges, and dropping the 10% most suspicious surviving edges recovers the whole 8.0 points the copies cost and 2.5 beyond it (Table 35). Told about the check, the attacker copies about one and a half more words per inserted edge, 524 where the minimal construction needed 136; the strongest score falls to 0.594 and that pruning removes 7 of the 263 inserted edges instead of 177. The check is one more price rise of a word or two per edge.

Which side of normal. Which tail the attack occupies is not fixed. Corroboration flags CLASP's edges as the *least* corroborated at a zero feature budget and as the *most* at 10%, moving monotonically through chance and crossing it at about 1% of the feature entries on Cora and 2% on CiteSeer (Fig. 4). A defender must fix the direction in advance: pruning the least corroborated tenth gains 1.8 pp against structure alone on Cora and loses 2.9 at a 10% budget. The sign of a one-sided similarity test is therefore a free parameter for the attacker. Reading it two-sidedly recovers the signal but not the accuracy: at 10% the edges are still separable (0.710 on Cora, 0.791 on CiteSeer), yet pruning them changes little, because the loss there is feature poisoning rather than the edges (App. K).

Continuous features. Copying a word has no analogue when attributes are not binary, so we repeat the construction on PubMed with its TF-IDF weights left as they are and a cosine filter. Moving each inserted edge's lower-degree endpoint towards the other takes the share of PRBCD's edges the filter keeps from 49.8% to all of them at $\tau = 0.05$, for 0.58% of the feature matrix's mass, and the defended accuracy from 49.9 to 27.8% (App. M).

5.4 When blinding costs accuracy

Heterophilous graphs and dense labels. At a 10% feature budget, joint poisoning corrupts the filter's evidence on every graph we measured, but what that costs follows the damage the kept edges do. On the heterophilous graphs the filter keeps more of CLASP's inserted edges as the budget grows (actor 60% → 73%, chameleon 20% → 67%, squirrel 30% → 69%; Table 1), yet the defended loss stays within 2.4 pp of its structure-only value on actor and squirrel and falls on chameleon (Table 10): CLASP's structural attack costs an undefended GCN at most 8.7 pp there, and the filter already removes 79% and 83% of chameleon's and squirrel's clean edges, so blinding returns clean edges along with the attacker's. Dense labels do the same, not by removing the blinding but by removing the damage: under the geom-GCN split, which labels 44% and 48% of Cora's and CiteSeer's nodes, the filter still keeps 76% and 92% of the inserted edges, while the attack costs at most 4.2 and 5.4 pp, defended or not (Table 11). Where the kept edges are damaging, a heterophilous graph is no protection: PRBCD costs chameleon's GCN 52.7 pp (Table 5), the filter removes 89% of its edges and holds the defended loss to 7.1, and 1,093 copied words (3.7% of non-zeros) make it keep all of them, raising that loss to 36.2; on squirrel the same construction raises it from 27.3 to 40.8 (Table 9).

A large budget blinds a learned similarity defense. SimP-GCN (Jin et al., 2021) learns how far to trust a feature k-nearest-neighbour graph beside the given one, with a self-supervised loss that preserves feature similarity. Against structure alone it loses 8.7 and 6.7 pp on Cora and CiteSeer, where an undefended GCN loses 17.6 and 19.1 (Table 27). But it computes its kNN graph and similarity targets from the features it is given, and at a 10% feature budget it loses 50.0 and 47.8 pp, more than the undefended GCN's 40.9 and 46.5; built from the clean features, the same model loses 23.6 and 15.4. It also loses more than the undefended GCN on actor, and ends below it in accuracy on Cora (30.3 against 39.3%).

Learned evidence is the exception to the cheap forgery. The defenses broken in Sec. 5.3 all read the raw features. A STABLE-style purifier (a Deep Graph Infomax encoder and a cosine cut on its embeddings; App. G) and SimP-GCN learn their evidence first, and the cheap forgery does not buy them (App. H). On Cora the 136 copies that take GCN-Jaccard from keeping 47% of Metattack's inserted edges to all of them move the purifier from 61% to 62%, and three check-aware copies per edge reach 71%; on Cora and CiteSeer the copies never change either defense's accuracy by more than 2.8 points, and the sign of the change is not stable. Tightening the purifier's cut until it removes 96% of the inserted edges does not change this (App. J). What does blind both is a large budget: at

10% the purifier keeps 85% of CLASP's inserted edges on Cora from poisoned embeddings against 62% from clean ones (78 against 45% on CiteSeer), and 5.7 of its 45.1 points of loss are that blinding (Table 29). An attacker handed the purifier's architecture, seed and cut, buying words that raise its embedding cosine rather than its feature overlap, does reliably better than the copies: 1.67 to 2.47 points across four cases for 0.22 to 0.85% of the feature entries (App. I). On Cora that is well short of the 5.7 points a 10% budget buys; on CiteSeer, where the 10% budget buys only 2.1, it is most of it. Learned evidence thus raises the attacker's price well above what raw-feature rules charge, but it is not a wall. Pro-GNN keeps a small advantage over the undefended GCN, 3.0 pp against structure alone, and is where CLASP's structural component loses to Metattack (18.4 against 14.6 pp on Cora; Table 28). Self-supervised purifiers (Lee and Park, 2025) remain untested.

Two more countermeasures. Ignoring each edge's k most helpful shared words, $(i - k)/(U - k) \geq \tau$, stops the copies as built but not an attacker who knows k, who keeps every inserted edge at every k we tried, for 340 copies instead of 136 at $k = 1$ (Tables 19 and 20). A median-aggregation GCN, which reads no features, loses 46.2 pp at a 10% budget against 40.9 undefended (Table 22). Both findings reproduce under the $10/10/80$ protocol (App. E).

Ablations. Two ablations locate CLASP's advantage (App. R). Class conditioning is not it: the pool the rule selects beats uniform sampling on Cora alone and is below it on all three heterophilous graphs. The shared objective is: at equal budgets it wins 17 of 24 comparisons against running structure and features in sequence.

6 Discussion and conclusion

A similarity filter that reads raw features fails in two ways once the attacker controls them. Its evidence can be forged cheaply: on the citation graphs about one copied word per inserted edge makes GCN-Jaccard keep every edge an attack inserts, GNNGuard's stricter threshold costs about two, and a higher threshold, a rule that discounts shared words, or an edge-aware check only moves the price, which the defender pays in clean edges. And it cannot stop feature poisoning, which at the larger budgets does most of the damage. Defenses that learn their evidence first, a STABLE-style purifier and SimP-GCN, are the exception to the first failure: the copies that blind the raw-feature rules barely change their accuracy, and only a budget large enough to change what the nodes say blinds them substantially. We would take a defense in this direction, though a joint attack already reaches the budget that blinds it.

7 Limitations

Edge-level measurements are single attack draws with three victim seeds, except on Cora and CiteSeer, where three more attack seeds repeat them. The similarity defenses measured edge by edge are Jaccard and cosine thresholds, GNNGuard, and a STABLE-style purifier: reimplemented, without the recovery step, so it stands for that design and not for STABLE's reported numbers. Every feature matrix here is a bag of words, binary except on PubMed, so the forgery is priced on sparse count-like attributes; on learned or dense embeddings the propositions still apply but the price in anything a human would notice is untested. Five unsupervised feature-side detectors are tested; on the two graphs where the attack costs the most accuracy the best stays at or below 0.616 AUC (App. O). A supervised detector is untested and may well work. On PubMed the default Jaccard threshold is inert and every gradient-based attack drives the GCN below chance, so its public-split accuracies rank attacks poorly; App. M compares a filter against itself there instead. The 5% and 10% budgets change node content, moving a clean-trained MLP's prediction on about 30% of the flipped test nodes against 3 to 14% for the copies. The joint attack's advantage is its extra budget: splitting one between edges and features does worse than structure alone (App. B). On chameleon and squirrel GCN-Jaccard removes about 80% of the clean edges before any attack, and both have duplicate nodes that leak across splits (Platonov et al., 2023); PRBCD and GRBCD are transferred evasion attacks; and PubMed is the largest graph attacked.

Reproducibility Statement

Every results table except the dataset table is generated by a script directly from stored result files, and the dataset table is checked against its file. A second script checks every number in the prose, the captions and the hand-written tables against the stored files and fails if any number in the prose goes unchecked. Both ship with our artifacts, together with every poisoned graph the paper scores: all feature budgets and variants on the six graphs and both splits, the attack-seed replication, and the graphs behind Tables 5, 6 and 14. The attack is deterministic on CPU for a fixed seed: regenerating the graphs of Table 10 reproduced every stored pruning count and accuracy exactly, and rerunning the PubMed attack reproduced its stored drop. Baselines use public implementations: DeepRobust for Metattack, PGD and MinMax, PyTorch Geometric (Fey and Lenssen, 2019) for PRBCD and GRBCD, and the authors' code for Metacon and GNNGuard. Our copy of Metacon carries four memory patches that reproduce the released code's output exactly (App. T). All experiments ran on one 2-core CPU machine with 8 GB of memory.

Ethics Statement

This work studies training-time adversarial manipulation of public benchmark graphs for robustness analysis. Graph poisoning is dual-use: similar methods could be applied to deployed relational systems when an attacker can alter topology or attributes. We therefore emphasize controlled benchmark evaluation, explicit perturbation budgets, and defensive interpretation rather than operational deployment guidance.

AI Use Statement

Large language models were used only in a limited editorial capacity during the preparation of this manuscript, primarily for occasional language polishing, grammar correction, and improving readability. They were not used to generate or design the research methodology, conduct experiments, implement the proposed method, analyze or interpret experimental results, formulate mathematical claims, or generate research findings. All experiments, analyses, methodological decisions, and scientific conclusions were performed and verified by the authors. The authors reviewed and take full responsibility for the entire contents of this submission.

References

Mohammad Sadegh Akhondzadeh, Soroush H. Zargarbashi, Jimin Cao, and Aleksandar Bojchevski. EvA: Evolutionary attacks on graphs. In *ICLR*, 2026.

Zulfikar Alom, Tran Gia Bao Ngo, Murat Kantarcioglu, and Cuneyt Gurcan Akcora. GOttack: Universal adversarial attacks on graph neural networks via graph orbits learning. In *ICLR*, 2025.

Yongqiang Chen, Han Yang, Yonggang Zhang, Kaili Ma, Tongliang Liu, Bo Han, and James Cheng. Understanding and improving graph injection attack by promoting unnoticeability. In *ICLR*, 2022.

Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. All you need is low (rank): Defending against adversarial attacks on graphs. In *WSDM*, 2020.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. In *NeurIPS*, 2021.

Lukas Gosch, Simon Geisler, Daniel Sturm, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Adversarial training for graph neural networks: Pitfalls, solutions, and new directions. In *NeurIPS*, 2023a.

Lukas Gosch, Daniel Sturm, Simon Geisler, and Stephan Günnemann. Revisiting robustness in graph machine learning. In *ICLR*, 2023b.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *KDD*, 2020.

Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node similarity preserving graph convolutional networks. In *WSDM*, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Woohyun Lee and Hogun Park. Self-supervised adversarial purification for graph neural networks. In *ICML*, pages 33715–33735, 2025.

Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Reliable representations make a stronger defender: Unsupervised structure refinement for robust GNN. In *KDD*, 2022.

Kuan Li, Yang Liu, Xiang Ao, and Qing He. Revisiting graph adversarial attack and defense from a data distribution perspective. In *ICLR*, 2023.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. DeepRobust: A PyTorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? In *NeurIPS*, 2022.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *ICLR*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *ICLR*, 2023.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021. doi: 10.1093/comnet/cnab014.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008. doi: 10.1609/aimag.v29i3.2157.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *IJCAI*, 2019.

Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *IJCAI*, pages 3961–3967, 2019.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48, 2016.

Kanghoon Yoon, Yeonjun In, Namkyeong Lee, Kibum Kim, and Chanyoung Park. Debiased graph poisoning attack via contrastive surrogate objective. In *CIKM*, 2024. doi: 10.1145/3627673.3679686.

Shen You, Kai Zhou, Zhongshen Li, Kay Chen Tan, Qiuzhen Lin, Xiangtao Li, and K.-C. Wong. SOPA: Sensitivity-oriented poisoning attack for self-supervised graph embedding model via bilevel evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 30(3):1108–1122, 2026.

Xiang Zhang and Marinka Zitnik. GNNGuard: Defending graph neural networks against adversarial attacks. In *NeurIPS*, 2020.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *KDD*, 2019.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*, 2018.

A ALGORITHM AND SETTINGS

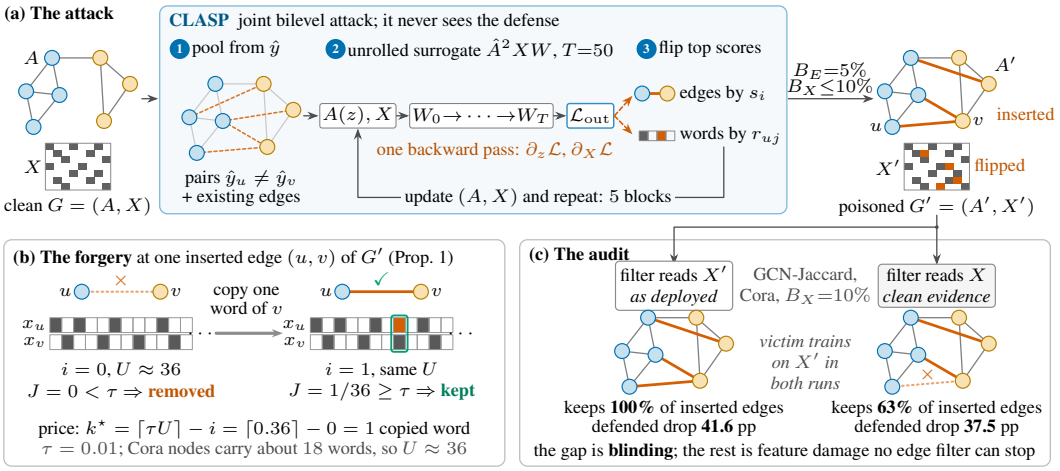

Figure 3: CLASP and the measurement. (a) A clean surrogate supplies pseudo-labels, the structural search is restricted to a class-conditioned pool, and one unrolled bilevel objective scores edge and feature flips jointly, in five blocks; the attack never sees the defense. (b) The poisoned features X' are also the evidence a similarity-based defense consults: one copied word lifts an inserted edge past GCN-Jaccard's threshold (Prop. 1). (c) The audit runs the defense twice on the same poisoned graph and measures which edges it keeps (Cora, 10% feature budget, Table 1).

Algorithm 1 CLASP

Require: graph (A, X), training labels, edge budget B_E, feature budget B_X, $K = 5$ blocks, $T = 50$ inner steps

fit the linearised surrogate on (A, X); $\hat{y} \leftarrow$ training labels, surrogate predictions elsewhere
$h \leftarrow$ edge homophily of A (all labels in our runs; $\hat{y}$ gives the same choice on all six graphs, Sec. 3)
$\mathcal{P} \leftarrow$ for each class c, pairs (u, v) with $\hat{y}_u = c$ and $\hat{y}_v \neq c$ if $h > 0.5$, $\hat{y}_v = c$ otherwise; add every existing edge
$z_p \leftarrow 1$ if $p \in E$ and 0 otherwise, for $p \in \mathcal{P}$
for $k = 1, \ldots, K$ **do**
    unroll T momentum steps of the surrogate on $(A(z), X)$, differentiable in z and X
    $\mathcal{L} \leftarrow$ cross-entropy on unlabelled nodes against $\hat{y}$
    $s \leftarrow \text{zscore}\big(\partial\mathcal{L}/\partial z \odot (1 - 2z)\big)$; exclude chosen pairs and removals that would isolate a node
    flip the $\lfloor B_E/K \rfloor$ highest-scoring pairs
    $r \leftarrow \partial\mathcal{L}/\partial X \odot (1 - 2X)$; flip the $\lfloor B_X/K \rfloor$ highest-scoring unflipped entries
end for
restore one deleted edge for any node left isolated
return $(A(z), X)$

Both gradients in a block come from the same backward pass through the same unrolled surrogate. Table 3 lists every setting.

Table 3: Settings used throughout.

Item	Setting
Surrogate	two-layer linearised GCN $\hat{A}^2XW$, $T = 50$ inner steps, learning rate 0.1, momentum 0.9, initialised from the attack seed
Schedule	5 blocks; edges and features flip in the same blocks; the edge budget's remainder after division by 5 is left unspent
Candidate pool	20 sampled partners per node below $4{,}000$ nodes, 8 below $10{,}000$, 4 above, plus all existing edges; $58{,}936$ pairs on Cora (1.6% of all pairs) and $70{,}594$ on CiteSeer
Pool rule	cross-class pairs when edge homophily (all labels in our runs; pseudo-labels give the same choice) exceeds 0.5, same-class pairs otherwise
Victims	PyG GCN, hidden 64, dropout 0.5, Adam with learning rate 0.01 and weight decay 5×10^{-4}, 200 epochs, test accuracy at the best validation epoch; the adapter victims add a rank-8 residual low-rank adapter after each layer of a GCN or an 8-head GAT, trained from scratch
GCN-Jaccard	DeepRobust, threshold $\tau = 0.01$ (removed iff $J < \tau$), hidden 64; a featureless node receives its own indicator column
GCN-SVD, RGCN, Pro-GNN	DeepRobust; SVD rank 15; Pro-GNN 200 epochs
Metattack	DeepRobust, Meta-Self ($\lambda = 0$), without its degree-likelihood check (`ll_constraint=False`); CLASP has no such check either
PRBCD, GRBCD	PyTorch Geometric, attacking a trained GCN on the test nodes (PRBCD: 125 epochs, block size $250{,}000$); the perturbed graph is then used for poisoning

B CONTROLS

Design alternatives, on the final configuration. Each change to the structural attack is paired against the reported configuration over ten victim seeds. A four-times finer schedule (20 blocks) is much stronger: 24.48 against 17.85 pp on Cora and 23.50 against 18.97 on CiteSeer ($p < 10^{-4}$ for both), at 114 and 173 s of attack time. A margin outer loss is stronger too, modestly on Cora (19.14 pp, $p = 0.019$) and markedly on CiteSeer (25.46 pp, $p < 10^{-4}$). A three-surrogate ensemble is weaker on Cora (15.72 pp, $p < 10^{-4}$) and ran out of memory on CiteSeer. We keep five blocks and cross-entropy throughout, so the paper's CLASP numbers understate what these changes give.

Budget response. Degradation grows with the structural budget on five of the six graphs, so the 5% headline is not an isolated operating point (Table 18); on actor it does not, because the structural attack costs at most 1.1 pp there at any budget.

Where the flips land. At a 0.25% budget CLASP's feature flips touch 72 Cora nodes, 1.7 per node on average and at most 4; at 10% they touch $1{,}469$ nodes, 3.3 per node and at most 95. The camouflage copies of Table 2 touch 132 nodes of Metattack's graph, at most two words each.

Decomposition, joint versus sequential, and protected labels. Table 17 holds the final-configuration decomposition. With the feature budget fixed, adding the structural attack raises the drop by 16.53 and 13.15 pp on Cora and by 7.95 and 7.86 on CiteSeer at 5% and 10% ($p < 10^{-4}$ for all four), so the joint attack is not a feature attack with a structural label. On chameleon it is the other way round: the joint attack is weaker than structure alone (4.71 and 5.24 against 7.43 pp), so on that graph the feature budget buys blinding at the cost of damage. Protecting the labelled nodes' features lowers the joint attack's damage by at most 4.0 pp (37.52 against 41.56 on Cora at 10%) but removes most of its camouflage: on Cora the filter then keeps 71% of inserted edges at 10% instead of all of them.

Matched total budget. Assigning half the original edge-flip count to structure and the rest to feature bits gives 12.80 pp on Cora and 16.82 pp on CiteSeer, *below* pure structure (17.56/19.46), and it stays below pure structure on all six graphs (Table 26), so the joint attack's advantage comes from an additional budget the structure-only baselines were not given. An edge flip and a feature-bit flip need not have equal semantic cost, so we treat this as a diagnostic rather than a universal cost-matched threat model.

C PubMed degree audit

For the PubMed entry of Table 24, measured on the stored poisoned graph of that run: maximum degree is essentially unchanged ($171 \rightarrow 170$) and mean degree moves $4.50 \rightarrow 4.60$, so the perturbation is not crude hub construction. The 60 labelled nodes absorb 404 of 3,234 inserted edge endpoints (12.5%) while being 0.30% of the graph, a $41\times$ over-representation; their mean degree rises $4.95 \rightarrow 8.07$, and six of the ten largest degree gainers are training nodes.

D Additional tables

Table 4: The attack and defense grid on all six graphs: accuracy drop (points) at a 5% structural budget, five victim seeds per cell, each drop taken against that victim's own clean accuracy on the same graph. Joint rows additionally flip the stated fraction of *non-zero* feature entries. Jaccard, SVD and RGCN are DeepRobust's implementations. PubMed's features are binarized. GCN-SVD and RGCN build dense $n \times n$ matrices and were killed by the out-of-memory killer on PubMed (*oom*).

Dataset	Attack	GCN	GCN+Ad.	GAT+Ad.	Jaccard	SVD	RGCN
Cora	Random	1.26	2.16	2.48	1.12	0.50	2.20
	DICE	0.54	1.08	1.84	0.66	2.40	1.48
	Structure only	17.56	17.08	14.96	10.46	3.90	16.64
	Joint 5% X_{nz}	34.66	35.58	33.88	35.50	16.50	36.54
	Joint 10% X_{nz}	41.06	41.38	38.16	41.62	22.34	41.54
CiteSeer	Random	2.12	1.28	3.44	0.36	0.66	2.22
	DICE	2.00	0.60	4.22	1.58	1.94	2.60
	Structure only	19.46	17.32	14.50	15.02	1.96	12.80
	Joint 5% X_{nz}	42.28	39.66	39.24	41.82	25.40	31.46
	Joint 10% X_{nz}	47.80	46.16	41.94	46.28	28.28	33.34
PubMed	Random	1.80	3.08	0.68	2.00	oom	oom
	DICE	1.70	2.66	1.20	1.46	oom	oom
	Structure only	54.28	51.80	39.50	52.56	oom	oom
	Joint 5% X_{nz}	57.30	56.78	52.38	50.84	oom	oom
	Joint 10% X_{nz}	57.88	57.70	43.14	53.68	oom	oom
actor	Random	0.42	−0.68	−0.04	−0.64	0.09	0.29
	DICE	1.43	0.63	0.62	0.43	0.22	1.58
	Structure only	0.75	0.55	0.45	1.25	0.58	0.28
	Joint 5% X_{nz}	2.46	1.07	1.63	3.36	2.21	0.07
	Joint 10% X_{nz}	2.88	1.86	2.80	4.46	3.25	0.14
chameleon	Random	4.74	6.58	6.54	0.96	3.90	4.17
	DICE	3.51	4.21	2.24	0.88	1.10	2.02
	Structure only	7.94	7.98	6.89	−0.18	3.38	4.91
	Joint 5% X_{nz}	5.53	5.92	6.84	−3.86	2.98	5.35
	Joint 10% X_{nz}	5.35	7.28	5.13	−6.89	1.67	1.36
squirrel	Random	5.17	4.82	7.51	1.96	1.29	2.75
	DICE	5.15	5.09	9.11	3.07	0.21	4.40
	Structure only	16.23	15.75	13.70	6.38	3.32	6.47
	Joint 5% X_{nz}	16.89	17.71	14.43	5.94	2.23	7.30
	Joint 10% X_{nz}	17.62	18.06	14.64	6.80	1.58	6.13

Table 5: Structure-only poisoning at a 5% edge budget on three homophilous and three heterophilous graphs: accuracy drop (points) of a GCN, five victim seeds, one attack draw each (Table 7 regenerates CLASP and PRBCD with independent attack seeds on every graph). Metattack ran on Cora, CiteSeer and chameleon; by the n^2B rule fitted to those three it would take about a day on actor, 3.4 days on squirrel and 11 on PubMed. PubMed's features are binarized here, so its row differs from the TF-IDF runs of Table 24, which are also the only uniform-pool runs on that graph, hence the n/a. On the heterophilous graphs CLASP's rule samples same-class pairs; the uniform-pool row replaces the rule by PRBCD's sampling, and its Cora, CiteSeer and squirrel cells come from that sweep, which reproduces every other CLASP cell here exactly. Table 6 adds PGD and MinMax with each attack's time and peak memory, measured one attack per process.

	homophilous			heterophilous		
Attack	Cora	CiteSeer	PubMed	actor	chameleon	squirrel
Random	1.26	2.12	1.80	0.42	4.74	5.17
DICE	0.54	2.00	1.70	1.43	3.51	5.15
GRBCD	6.08	10.12	55.12	14.72	31.80	34.64
PRBCD	15.84	16.56	57.92	16.51	52.72	46.86
Metattack	13.26	15.18	n/a	n/a	12.28	n/a
CLASP	17.56	19.46	54.28	0.75	7.94	16.23
CLASP, uniform pool	14.60	19.24	n/a	2.25	8.29	20.02

Table 6: Every structural attack on every graph where it ran: same 5% edge budget, same victim (a GCN, five victim seeds), one attack draw each. Time covers everything a method needs, including training its own surrogate. Peak memory is resident set size above baseline, measured in a *separate* process per attack, because within one process the allocator does not return freed pages and every attack after the first would be understated. p is a paired t-test against CLASP over the shared victim seeds, so it measures victim-training noise for one graph per attack; Table 7 regenerates CLASP, PRBCD and Metattack with independent attack seeds. PGD, MinMax and Metattack hold dense $n \times n$ matrices: none of them was run on PubMed, where PGD's perturbation vector alone has 194 million entries, and Metattack not on actor or squirrel, where its measured $n^2 B$ scaling predicts about a day and 3.4 days. CPU measurements; absolute memory differs on a GPU, but the ordering reflects each method's working set.

Dataset	Attack	Drop (pp)	p vs CLASP	Time (s)	Peak mem (MB)
Cora	Random	1.26	<0.0001	0.0	0.1
	DICE	0.54	<0.0001	0.0	0.1
	MinMax	0.36	<0.0001	489.8	724.0
	PGD	2.02	<0.0001	278.8	665.2
	GRBCD	6.08	<0.0001	69.3	537.8
	Metattack	13.26	0.0006	2090.0	1208.5
	PRBCD	15.84	0.0137	100.3	546.7
	CLASP	**17.56**	n/a	**17.4**	**1412.4**
CiteSeer	Random	2.12	<0.0001	0.0	1.2
	DICE	2.00	<0.0001	0.0	2.4
	MinMax	2.10	<0.0001	887.6	980.7
	PGD	10.38	0.0001	498.8	981.2
	GRBCD	10.12	<0.0001	75.8	537.8
	Metattack	15.18	0.0006	3116.7	1191.7
	PRBCD	16.56	0.0022	118.2	545.3
	CLASP	**19.46**	n/a	**26.5**	**2423.8**
PubMed	Random	1.80	<0.0001	0.1	13.0
	DICE	1.70	<0.0001	0.3	15.0
	GRBCD	55.12	0.6546	73.7	655.1
	PRBCD	57.92	0.0247	104.9	661.8
	CLASP	**54.28**	n/a	**33.6**	**2172.0**
actor	Random	0.42	0.2980	0.0	7.1
	DICE	1.43	0.0963	0.1	7.3
	MinMax	2.00	0.0158	8594.5	3454.7
	PGD	6.53	<0.0001	4357.2	3339.5
	GRBCD	14.72	0.0007	56.9	588.2
	PRBCD	16.51	0.0002	88.6	629.4
	CLASP	**0.75**	n/a	**19.0**	**2686.7**
chameleon	Random	4.74	0.0005	0.1	3.5
	DICE	3.51	0.0010	0.1	6.2
	MinMax	9.61	0.0833	276.7	477.2
	PGD	8.68	0.2946	151.2	436.0
	GRBCD	31.80	<0.0001	51.6	593.6
	Metattack	12.28	0.0042	4815.3	742.2
	PRBCD	52.72	<0.0001	80.8	594.7
	CLASP	**7.94**	n/a	**19.7**	**1509.5**
squirrel	Random	5.17	<0.0001	0.4	29.1
	DICE	5.15	<0.0001	0.4	36.7
	MinMax	12.43	0.0005	2850.1	1708.6
	PGD	14.89	0.0312	1666.6	1647.9
	GRBCD	34.64	<0.0001	117.0	930.7
	PRBCD	46.86	<0.0001	150.3	924.3
	CLASP	**16.23**	n/a	**56.0**	**4495.3**

Table 7: Attack-seed replication of the structural comparison on all six graphs: each attack is regenerated with independent attack seeds, each graph is scored with the five victim seeds of Table 6, and p is Welch's t-test on the per-seed mean drops against CLASP. CLASP and PRBCD have ten attack seeds on Cora and CiteSeer and five on the other four graphs, except on squirrel, where CLASP's fifth run was killed by the out-of-memory killer at about 6 GB. Metattack takes 35 to 52 minutes per run on Cora and CiteSeer and hours on chameleon, so it has three attack seeds on Cora; rows marked * show the single draw of Table 6.

Dataset	Attack	Seeds	Drop (pp), mean	std	p vs CLASP
Cora	CLASP	10	16.94	2.24	n/a
	PRBCD	10	16.56	0.67	0.6164
	Metattack	3	9.31	1.23	0.0002
CiteSeer	CLASP	10	19.83	1.41	n/a
	PRBCD	10	16.81	0.77	$<$0.0001
	Metattack*	1	15.18	n/a	n/a
PubMed	CLASP	5	61.02	3.93	n/a
	PRBCD	5	57.86	0.83	0.1474
actor	CLASP	5	0.83	0.22	n/a
	PRBCD	5	16.40	0.49	$<$0.0001
chameleon	CLASP	5	7.56	0.90	n/a
	PRBCD	5	52.14	0.29	$<$0.0001
	Metattack*	1	12.28	n/a	n/a
squirrel	CLASP	4	16.66	0.80	n/a
	PRBCD	5	46.45	0.70	$<$0.0001

Table 8: Every feature budget on all six graphs (Table 1 shows 0 and 10%): CLASP's stored graphs at a 5% edge budget, GCN-Jaccard at DeepRobust's default $\tau = 0.01$, except on PubMed, where that threshold is nearly inert and the filter is shown at $\tau = 0.1$ (Table 13 has the other thresholds). *Kept* is the share of the attack's inserted edges that survive the filter when it sees the poisoned features and when it is given the clean ones. Drops are in points against each model's clean accuracy, over three victim seeds: the undefended GCN, GCN-Jaccard as deployed, and GCN-Jaccard whose filter is given the clean features while its GCN still trains on the poisoned ones. Cora, CiteSeer and PubMed use the public split, the heterophilous graphs the geom-GCN split and the same-class pool CLASP's rule selects there.

		inserted edges kept		accuracy drop (pp)		
Dataset	Flips (% X_{nz})	poisoned ev.	clean ev.	undefended	Jaccard	clean ev.
Cora	0	61%	61%	17.6	10.3	10.3
	0.25	80%	64%	21.7	20.7	18.2
	1	92%	64%	25.1	26.8	21.1
	2	92%	63%	30.0	32.0	26.7
	5	97%	64%	34.1	35.2	31.7
	10	100%	63%	40.9	41.6	37.5
CiteSeer	0	74%	74%	19.1	14.9	14.9
	0.25	85%	77%	24.0	22.9	21.6
	1	94%	74%	31.8	30.4	27.2
	2	97%	69%	36.5	35.8	30.8
	5	100%	72%	41.0	40.4	37.0
	10	100%	72%	46.5	45.7	42.6
PubMed, $\tau = 0.1$	0	35%	35%	55.9	23.1	23.1
	0.25	40%	35%	66.9	51.0	48.5
	1	47%	35%	57.5	60.8	48.6
	2	50%	36%	49.7	64.7	47.2
	5	56%	38%	52.9	54.5	61.3
	10	66%	39%	55.3	52.8	51.8
actor	0	60%	60%	0.7	1.4	1.4
	0.25	62%	59%	0.9	1.1	1.1
	1	65%	59%	0.9	1.2	1.1
	2	65%	57%	1.4	1.2	0.9
	5	67%	55%	2.4	3.2	2.3
	10	73%	52%	2.5	3.8	2.7
chameleon	0	20%	20%	7.7	0.0	0.0
	0.25	21%	21%	6.8	1.5	1.4
	1	20%	20%	7.9	2.5	2.0
	2	21%	20%	6.7	1.5	3.2
	5	41%	17%	5.8	−3.1	1.4
	10	67%	16%	5.7	−6.5	−1.2
squirrel	0	30%	30%	16.5	6.4	6.4
	0.25	30%	30%	15.8	6.5	6.2
	1	30%	30%	15.7	5.8	7.1
	2	30%	30%	16.8	5.8	6.5
	5	53%	29%	16.6	6.1	5.8
	10	69%	28%	17.4	6.8	6.1

Table 9: The camouflage baseline of Table 2 for every stored structural attack on all six graphs: the fewest copied words that lift every inserted edge to the filter's Jaccard threshold (Prop. 1); no other feature changes. The threshold is 0.01; PubMed is shown at 0.01, where the filter is nearly inert, and at 0.1. Metattack ran on Cora, CiteSeer and chameleon only (Table 6), so it appears on those three graphs. Drops are in points against each model's clean accuracy, three victim seeds.

				inserted edges kept		Jaccard drop (pp)	
Dataset	Attack	Copies	% X_{nz}	before	after	before	after
Cora	Metattack	136	0.28	47%	100%	7.6	15.7
	PRBCD	86	0.17	67%	100%	10.9	14.9
	CLASP (structure)	96	0.20	61%	100%	10.3	18.6
CiteSeer	Metattack	57	0.05	74%	100%	12.0	13.2
	PRBCD	55	0.05	76%	100%	13.8	16.8
	CLASP (structure)	57	0.05	74%	100%	14.9	19.0
PubMed, $\tau = 0.01$	PRBCD	10	<0.01	100%	100%	57.4	57.7
	CLASP (structure)	17	<0.01	99%	100%	51.8	51.9
PubMed, $\tau = 0.1$	PRBCD	4,566	0.46	42%	100%	30.0	55.3
	CLASP (structure)	4,316	0.44	35%	100%	23.1	53.9
actor	PRBCD	536	1.31	57%	100%	8.9	15.6
	CLASP (structure)	440	1.07	60%	100%	0.9	1.3
chameleon	Metattack	1,248	4.28	15%	100%	3.7	13.9
	PRBCD	1,093	3.75	11%	100%	7.1	36.2
	CLASP (structure)	959	3.29	20%	100%	3.1	2.6
squirrel	PRBCD	5,460	5.84	34%	100%	27.3	40.8
	CLASP (structure)	3,593	3.84	30%	100%	7.6	11.8

Table 10: The blinding sweep on the five binary-feature graphs. *Prunes* is the share of edges GCN-Jaccard removes from the *clean* graph and *cost* the clean accuracy it gives up to do so. Pruning counts mix inserted and clean edges; the last column is the share of CLASP's inserted edges the filter keeps at feature budgets of 0 and 10%, from the edge-level analysis of Tables 1 and 11. Cora and CiteSeer use the public split (3.6 to 5.2% of nodes labelled), the other three the geom-GCN split (48%).

		clean filter		edges pruned			defended drop (pp)			inserted kept
Dataset	h	prunes	cost	0%	5%	10%	0%	5%	10%	0% → 10%
Cora	0.81	10.8%	2.2	670	547	432	10.3	35.2	41.6	61%→100%
CiteSeer	0.74	2.6%	−0.2	174	122	112	14.9	40.4	45.7	74%→100%
actor	0.22	47.6%	−1.5	13,117	13,928	15,738	1.4	3.2	3.8	60%→73%
squirrel	0.22	82.9%	6.8	166,219	58,188	28,539	6.4	6.1	6.8	30%→69%
chameleon	0.23	79.4%	15.1	25,703	18,106	9,385	0.0	−3.1	−6.5	20%→67%

Table 11: The measurements of Table 8 in two other settings ($\tau = 0.01$, 5% edge budget, three victim seeds). Top: Cora and CiteSeer under the first geom-GCN split, which labels 44% and 48% of their nodes. Bottom: the heterophilous graphs with uniform block sampling (PRBCD's) in place of the same-class pool CLASP's rule selects there. *Kept*: share of inserted edges the filter keeps given the poisoned and the clean features. Drops are in points against each model's clean accuracy; *clean ev.* gives the filter the clean features while its GCN trains on the poisoned ones. *oom*: the attack was killed by the out-of-memory killer at about 6 GB.

			inserted edges kept		accuracy drop (pp)		
Graph	Setting	Flips (%)	poisoned ev.	clean ev.	undefended	Jaccard	clean ev.
Cora	geom split	0	56%	56%	1.2	1.4	1.4
		0.25	55%	55%	1.1	0.9	1.1
		5	68%	61%	2.3	4.2	1.8
		10	76%	59%	3.3	3.8	2.4
CiteSeer	geom split	0	68%	68%	0.9	−0.3	−0.3
		0.25	69%	68%	1.0	0.1	0.1
		5	78%	67%	4.0	3.9	3.6
		10	92%	68%	4.2	5.4	4.2
actor	uniform pool	0	55%	55%	2.5	1.8	1.8
		5	64%	57%	2.3	2.9	1.9
		10	68%	57%	3.3	4.9	3.2
chameleon	uniform pool	0	6%	6%	8.4	4.1	4.1
		5	26%	7%	7.6	1.2	5.0
		10	54%	7%	8.7	−1.8	1.6
squirrel	uniform pool	0	oom	oom	oom	oom	oom
		5	oom	oom	oom	oom	oom
		10	oom	oom	oom	oom	oom

Table 12: GNNGuard, run from its authors' code with the settings of their Metattack experiments, on the stored graphs of Tables 1 and 2 (5% edge budget, three victim seeds). GNNGuard drops an edge whose endpoints' cosine similarity is below 0.1 and reweights the rest; *kept* is the share of inserted edges above 0.1 at its first layer, given the poisoned and the clean features. Drops are in points against GNNGuard's own clean accuracy; *clean ev.* computes the first layer's weights from the clean features while the network propagates the poisoned ones. The structural attack whose graph is camouflaged is Metattack on Cora and CiteSeer and PRBCD on the heterophilous graphs. PubMed is not run: the authors' implementation builds a dense $n \times n$ similarity matrix in every epoch, which does not fit there.

			inserted edges kept		GNNGuard drop (pp)	
Dataset	Graph	Flips (%)	poisoned ev.	clean ev.	as deployed	clean ev.
Cora	CLASP, structure	0	15%	15%	4.0	4.0
	CLASP, joint	0.25	36%	17%	13.3	9.4
	CLASP, joint	5	72%	17%	35.2	24.2
	CLASP, joint	10	89%	18%	40.3	30.6
	Metattack	0	14%	14%	4.1	4.1
	+ copies for Jaccard	0.28	23%	14%	4.7	4.3
	+ copies for cosine	0.86	100%	14%	12.7	4.5
	clean accuracy 70.3%					
CiteSeer	CLASP, structure	0	10%	10%	3.0	3.0
	CLASP, joint	0.25	17%	9%	11.3	8.9
	CLASP, joint	5	73%	9%	35.5	30.7
	CLASP, joint	10	87%	10%	41.1	36.3
	Metattack	0	11%	11%	2.2	2.2
	+ copies for Jaccard	0.05	11%	11%	2.6	1.9
	+ copies for cosine	0.48	100%	11%	10.6	3.0
	clean accuracy 62.7%					
chameleon	CLASP, structure	0	3%	3%	0.1	0.1
	CLASP, joint	5	18%	2%	−3.3	−0.5
	CLASP, joint	10	39%	1%	−7.5	1.1
	PRBCD	0	2%	2%	0.4	0.4
	+ copies for cosine	7.48	100%	2%	30.7	9.1
	clean accuracy 44.3%					
actor	CLASP, structure	0	59%	59%	0.3	0.3
	CLASP, joint	5	66%	55%	1.8	0.5
	CLASP, joint	10	73%	52%	4.6	2.6
	PRBCD	0	55%	55%	8.0	8.0
	+ copies for cosine	1.46	100%	55%	13.2	8.3
	clean accuracy 31.3%					
squirrel	CLASP, structure	0	4%	4%	0.0	0.0
	CLASP, joint	5	18%	4%	0.1	0.5
	CLASP, joint	10	35%	4%	−1.2	1.8
	PRBCD	0	4%	4%	4.2	4.2
	+ copies for cosine	19.57	100%	4%	32.7	14.3
	clean accuracy 35.3%					

Table 13: PubMed (binarized features) at several Jaccard thresholds, on CLASP's PubMed graphs (Sec. 5.2). *Pruned*: clean edges the filter removes from the clean graph. *Kept*: share of inserted edges the filter keeps, by feature budget (% of non-zeros), given the poisoned features, and at 10% given the clean ones. Drops were measured at the two larger thresholds only, in points against the filter's clean accuracy, three victim seeds; *clean ev.* gives the filter the clean features. The two smaller thresholds were not run for accuracy (n/a): the filter keeps 97% or more of the inserted edges there before any feature flip, so there is no blinding left to measure.

		inserted edges kept						Jaccard drop (pp)		
τ	pruned	0	0.25	1	5	10	10, clean ev.	0	10	10, clean ev.
0.01	0.1%	99%	99%	100%	100%	100%	99%	n/a	n/a	n/a
0.02	0.2%	97%	98%	98%	99%	99%	97%	n/a	n/a	n/a
0.05	1.7%	80%	85%	89%	91%	92%	81%	47.9	53.7	58.0
0.1	14.3%	35%	40%	47%	56%	66%	39%	23.1	52.8	51.8

Table 14: Metacon (CIKM 2024) in its authors' own harness: their dataset loader and largest-connected-component setting, their random $10/10/80$ split, their linearised surrogate, and their per-dataset hyperparameters. Metattack and CLASP are run on the same graph, split and budget and scored by the same victim over five seeds. "Jac. pruned" is the number of edges GCN-Jaccard removes, which mixes inserted and clean edges (Sec. 5.2). Every attack here runs at a 1% edge budget, not the 5% used elsewhere. Metacon$_D$ could only be run at 20 inner steps rather than its authors' 100 (App. F), so Metattack is shown both at its default 100 and at 20, the matched comparison for Metacon$_D$.

Dataset	Attack	GCN drop (pp)	Jaccard drop (pp)	Jac. pruned
Cora	clean graph	GCN 83.88, Jaccard 82.38		548
	Metattack	2.23	0.82	568
	Metattack, 20 inner steps	1.24	0.37	571
	Metacon$_D$, 20 inner steps	1.07	0.57	565
	CLASP	1.41	0.75	563
	CLASP joint 5%	7.73	10.30	756
CiteSeer	clean graph	GCN 73.34, Jaccard 73.18		96
	Metattack	1.14	0.30	103
	Metattack, 20 inner steps	0.70	0.38	104
	Metacon$_D$, 20 inner steps	1.86	1.18	110
	CLASP	0.73	−0.14	102
	CLASP joint 5%	7.63	7.50	107

Table 15: Raising the Jaccard threshold, on all six graphs. Clean cost is the accuracy GCN-Jaccard gives up on the clean graph and the number of clean edges it prunes there. For each attacked graph: the defended drop in points and the share of inserted edges kept. *Camouflaged* graphs are a structural attack's graph with the word copies needed at $\tau = 0.01$ or $\tau = 0.05$ (Prop. 1): Metattack's on Cora and CiteSeer, PRBCD's on the other four. Drops here are against the undefended GCN's clean accuracy, so they include the filter's own clean cost. Three victim seeds.

		clean cost		CLASP, structure		camouflaged, 0.01		camouflaged, 0.05	
Dataset	τ	pp	pruned	drop	kept	drop	kept	drop	kept
Cora	0.01	2.2	572	12.5	61%	17.9	100%	17.7	100%
	0.02	2.2	572	12.5	61%	18.0	100%	17.7	100%
	0.03	4.5	1,049	9.4	28%	12.4	57%	20.6	100%
	0.05	5.5	1,605	9.4	16%	8.3	18%	23.4	100%
	0.1	11.6	3,349	12.4	1%	12.3	2%	13.0	2%
CiteSeer	0.01	−0.2	117	14.6	74%	12.9	100%	13.4	100%
	0.02	−0.6	418	11.2	42%	11.2	45%	15.1	100%
	0.03	0.9	568	7.9	32%	10.2	35%	16.1	100%
	0.05	2.8	1,229	3.8	10%	7.3	12%	20.4	100%
	0.1	5.8	2,617	6.3	0%	6.8	0%	7.2	0%
PubMed	0.01	0.6	39	52.4	99%	58.3	100%	58.0	100%
	0.02	0.2	102	52.0	97%	57.9	97%	58.1	100%
	0.03	0.2	207	53.0	93%	57.4	94%	58.1	100%
	0.05	−0.2	761	47.7	80%	53.5	83%	58.2	100%
	0.1	0.8	6,318	23.9	35%	30.8	42%	30.9	43%
actor	0.01	−1.5	12,696	−0.2	60%	13.6	100%	13.5	100%
	0.02	−1.5	12,696	−0.2	60%	13.6	100%	13.5	100%
	0.03	−1.6	12,700	−0.6	60%	14.0	100%	14.1	100%
	0.05	−1.4	12,894	−0.2	59%	12.9	94%	13.0	100%
	0.1	−2.8	15,984	−1.3	50%	5.7	54%	5.5	54%
chameleon	0.01	15.1	24,900	15.1	20%	49.7	100%	50.6	100%
	0.02	14.3	27,016	15.4	13%	46.1	83%	51.0	100%
	0.03	16.4	28,869	15.7	7%	43.3	65%	51.1	100%
	0.05	15.4	30,432	16.2	3%	38.2	40%	51.6	100%
	0.1	17.5	31,130	17.3	1%	29.8	17%	33.8	17%
squirrel	0.01	6.8	164,401	13.2	30%	46.7	100%	46.8	100%
	0.02	8.0	169,119	12.5	19%	44.9	73%	47.2	100%
	0.03	10.4	177,026	12.7	9%	39.7	44%	47.3	100%
	0.05	10.8	188,650	10.8	3%	28.7	16%	47.0	100%
	0.1	14.1	196,702	14.7	0%	20.5	2%	29.0	5%

Table 16: A cosine-similarity filter (DeepRobust's non-binary branch of GCN-Jaccard) on CLASP's graphs on all six graphs (PubMed's binarized features). Accuracy in %, and the share of inserted edges kept, three victim seeds.

		clean	structure only		joint 5%		joint 10%	
Dataset	τ	acc	acc	kept	acc	kept	acc	kept
Cora	0.01	77.9	67.6	61%	42.8	97%	36.3	100%
	0.05	77.1	68.7	44%	41.9	93%	35.5	97%
	0.1	74.4	71.1	15%	36.3	71%	30.6	89%
CiteSeer	0.01	68.9	54.0	74%	28.5	100%	23.2	100%
	0.05	69.0	58.8	39%	26.6	95%	21.1	98%
	0.1	66.1	64.0	10%	25.0	73%	19.9	87%
PubMed	0.01	78.2	26.4	99%	24.1	100%	25.6	100%
	0.05	78.4	28.4	95%	23.4	99%	25.6	99%
	0.1	79.0	32.7	80%	22.4	93%	24.9	94%
actor	0.01	32.2	30.9	60%	29.0	67%	28.4	73%
	0.05	32.1	31.0	60%	29.0	67%	28.5	73%
	0.1	32.1	30.5	59%	29.1	66%	28.0	73%
chameleon	0.01	50.7	50.2	21%	55.8	41%	56.7	67%
	0.05	50.6	49.8	11%	52.1	31%	56.7	59%
	0.1	47.7	47.6	3%	47.4	18%	51.7	39%
squirrel	0.01	42.3	35.0	31%	35.9	54%	34.9	70%
	0.05	40.3	34.6	17%	35.8	40%	34.6	59%
	0.1	37.6	36.7	4%	35.4	17%	34.7	35%

Table 17: Decomposition, joint versus sequential, and protected labels, on all six graphs: 5% edge budget, plain-GCN victim, ten victim seeds. *Extra* is the additional drop of the joint attack over the row's attack at the same budgets, with its paired t-test; negative means the row's attack is stronger.

		feature budget 5%		feature budget 10%	
Dataset	Attack	drop	joint extra (p)	drop	joint extra (p)
Cora	Structure only	17.85			
	Features only	18.28	16.53 (<0.0001)	28.41	13.15 (<0.0001)
	Structure, then features	27.10	7.71 (<0.0001)	31.28	10.28 (<0.0001)
	Features, then structure	36.58	−1.77 (0.0031)	38.35	3.21 (<0.0001)
	Joint (CLASP)	34.81		41.56	
	Joint, labelled nodes protected	35.09		37.52	
CiteSeer	Structure only	18.97			
	Features only	33.90	7.95 (<0.0001)	39.85	7.86 (<0.0001)
	Structure, then features	28.40	13.45 (<0.0001)	37.33	10.38 (<0.0001)
	Features, then structure	31.66	10.19 (<0.0001)	40.80	6.91 (<0.0001)
	Joint (CLASP)	41.85		47.71	
	Joint, labelled nodes protected	40.03		44.20	
PubMed	Structure only	51.10			
	Features only	50.29	3.34 (0.0653)	54.79	−1.63 (0.5828)
	Structure, then features	45.72	7.91 (0.0210)	44.72	8.44 (0.0410)
	Features, then structure	38.93	14.70 (0.0018)	36.54	16.62 (0.0006)
	Joint (CLASP)	53.63		53.16	
	Joint, labelled nodes protected	54.90		56.71	
actor	Structure only	0.80			
	Features only	−0.02	2.47 (<0.0001)	1.47	1.24 (0.0002)
	Structure, then features	2.28	0.16 (0.3467)	3.11	−0.39 (0.0358)
	Features, then structure	1.97	0.47 (0.0308)	3.35	−0.63 (0.1078)
	Joint (CLASP)	2.45		2.72	
	Joint, labelled nodes protected	2.78		2.80	
chameleon	Structure only	7.43			
	Features only	0.70	4.01 (<0.0001)	1.03	4.21 (<0.0001)
	Structure, then features	7.19	−2.48 (0.0032)	7.17	−1.93 (0.0022)
	Features, then structure	5.44	−0.72 (0.2007)	7.24	−2.00 (0.0034)
	Joint (CLASP)	4.71		5.24	
	Joint, labelled nodes protected	5.66		6.49	
squirrel	Structure only	16.03			
	Features only	−0.31	17.16 (<0.0001)	−0.61	17.95 (<0.0001)
	Structure, then features	15.75	1.10 (0.0033)	16.12	1.22 (0.0084)
	Features, then structure	13.00	3.85 (<0.0001)	11.21	6.13 (<0.0001)
	Joint (CLASP)	16.85		17.34	
	Joint, labelled nodes protected	17.41		17.11	

Table 18: Budget response of CLASP's structure-only attack on all six graphs: accuracy drop of a GCN (points, five victim seeds) at 1, 2, 5 and 10% of the edges, under the protocol of Table 5.

Dataset	1%	2%	5%	10%
Cora	2.84	6.70	17.56	30.20
CiteSeer	6.02	9.60	19.46	31.80
PubMed	31.20	40.08	54.28	60.14
actor	1.13	0.01	0.75	0.93
chameleon	2.06	3.90	7.94	9.08
squirrel	2.13	7.55	16.23	19.35

Table 19: A similarity rule that ignores the k most helpful shared words of an edge, $(i-k)/(U-k) \geq \tau$ with $\tau = 0.01$, on the stored graphs (Sec. 5.4). $k = 0$ is GCN-Jaccard and reproduces its numbers exactly. *Clean* is the filter's own accuracy on the clean graph and the clean edges it removes there; for each attacked graph, the defended accuracy and the share of inserted edges kept, three victim seeds. The camouflaged graph is Metattack's on Cora and CiteSeer and PRBCD's on chameleon, with the copies that pass $\tau = 0.01$ at $k = 0$.

		clean graph		CLASP, structure		CLASP, joint 10%		camouflaged	
Dataset	k	acc	pruned	acc	kept	acc	kept	acc	kept
Cora	0	77.9	572	67.6	61%	36.3	100%	62.3	100%
	1	75.0	1,472	70.5	24%	33.1	95%	70.8	21%
	2	71.2	2,570	70.0	8%	28.6	83%	70.6	8%
	3	67.9	3,497	67.1	1%	23.6	59%	66.3	3%
CiteSeer	0	68.9	117	54.0	74%	23.2	100%	55.7	100%
	1	68.7	459	58.6	39%	21.2	99%	57.7	41%
	2	67.0	966	62.2	17%	19.6	98%	60.3	20%
	3	65.3	1,507	64.4	6%	19.9	94%	61.9	8%
chameleon	0	50.7	24,900	50.7	20%	57.2	67%	16.2	100%
	1	49.5	29,443	51.5	8%	53.9	37%	37.3	6%
	2	51.2	30,724	52.1	3%	48.4	7%	40.7	1%
	3	50.7	31,162	50.1	2%	49.4	2%	41.0	0%

Table 20: The k-robust rule of Table 19 evaluated adaptively. *Copies for* $k = 0$: the camouflage built against plain Jaccard, which is what Table 19 scores. *Copies for this* k: the same construction told the value of k, which by Prop. 1 needs $\lceil \tau(U-k) \rceil + k - i$ copies per edge. *Kept* is the share of the structural attack's inserted edges the rule keeps and *acc* the defended accuracy, three victim seeds. The attack is Metattack's graph on Cora and CiteSeer and PRBCD's on chameleon.

		copies for $k = 0$		copies for this k		
Dataset	k	kept	acc	copies	kept	acc
Cora	0	100%	62.3	136	100%	62.3
	1	21%	70.8	340	100%	58.0
	2	8%	70.6	572	100%	52.1
	3	3%	66.3	824	100%	46.2
CiteSeer	0	100%	55.7	57	100%	55.7
	1	41%	57.7	190	100%	53.6
	2	20%	60.3	370	100%	49.6
	3	8%	61.9	572	100%	45.6
chameleon	0	100%	16.2	1,093	100%	15.5
	1	6%	37.3	2,244	100%	15.6
	2	1%	40.7	3,421	100%	15.3
	3	0%	41.0	4,590	100%	13.4

Table 21: The measurements of Tables 1 and 2 in the protocol most of the Metattack and Pro-GNN literature uses: DeepRobust's largest-connected-component setting with a random $10/10/80$ split, at the same 5% edge budget, with that harness's GCN victim (five seeds) and GCN-Jaccard (three). *Kept*: share of the attack's inserted edges the filter keeps, given the poisoned and the clean features. Drops are in points against each model's clean accuracy; *clean ev.* gives the filter the clean features while its GCN trains on the poisoned ones. + *copies* adds the fewest copied words that lift every inserted edge to $\tau = 0.01$.

		inserted edges kept		accuracy drop (pp)		
Dataset	Graph	poisoned ev.	clean ev.	undefended	Jaccard	clean ev.
Cora	CLASP, structure	65%	65%	6.1	3.2	3.2
	CLASP, joint 0.25%	70%	65%	6.6	4.2	3.2
	CLASP, joint 10%	95%	64%	27.7	30.0	24.7
	PRBCD	68%	68%	9.6	8.4	8.4
	+ copies	100%	68%	9.6	9.5	8.8
CiteSeer	CLASP, structure	79%	79%	5.5	3.5	3.5
	CLASP, joint 0.25%	80%	74%	5.9	5.0	4.4
	CLASP, joint 10%	99%	71%	21.9	21.9	19.8
	PRBCD	80%	80%	6.8	5.5	5.5
	+ copies	100%	80%	6.9	6.6	5.4

Table 22: A defense that never reads features: a two-layer GCN whose aggregation is the dimension-wise median over each node's closed neighbourhood (our implementation, in the spirit of MedianGCN), on the stored graphs at a 5% edge budget, three victim seeds. Drops are in points against its own clean accuracy. Metattack's graph and its camouflaged version differ only in features, so a feature-blind model must score them alike; the difference is a check. Metattack ran only on Cora, CiteSeer and chameleon (Table 6), so its two columns are n/a on the other three graphs.

		CLASP drop (pp)			Metattack drop (pp)	
Dataset	clean acc.	structure	joint 5%	joint 10%	plain	camouflaged
Cora	81.47	17.33	41.80	46.23	9.80	10.93
CiteSeer	70.50	16.13	46.03	48.70	8.70	7.67
PubMed	76.80	50.23	56.30	56.73	n/a	n/a
actor	29.01	1.05	3.68	4.74	n/a	n/a
chameleon	60.67	3.58	2.27	0.15	10.09	8.70
squirrel	45.47	11.72	11.18	11.40	n/a	n/a

Table 23: The edge-level measurement of Table 1 repeated with independent attack seeds: CLASP's structure-only and 10% joint graphs are regenerated with each seed and re-scored (three victim seeds each). Seed 0 is the draw the other tables report. *Kept*: share of that draw's inserted edges the filter keeps, given the poisoned features at each budget and, in the last kept column, given the clean ones at the 10% budget.

		inserted edges kept			accuracy drop (pp)		
Dataset	Seed	0	10	10, clean ev.	undef., 10	Jac., 0	Jac., 10
Cora	0	61%	100%	63%	40.9	10.3	41.6
	1	61%	99%	63%	37.4	7.5	39.4
	2	72%	100%	62%	40.3	12.2	41.8
	3	61%	100%	63%	38.3	9.9	39.7
	mean $\pm$ sd	64% $\pm$ 5	100% $\pm$ 0	63% $\pm$ 0	39.2	10.0	40.6
CiteSeer	0	74%	100%	72%	46.5	14.9	45.7
	1	66%	99%	65%	46.3	10.8	43.5
	2	77%	100%	73%	48.4	15.9	44.9
	3	74%	100%	66%	47.7	18.5	45.8
	mean $\pm$ sd	73% $\pm$ 5	100% $\pm$ 0	69% $\pm$ 4	47.2	15.0	45.0

Table 24: Candidate-pool sweep across the homophily range, every row from one script at the attack configuration used throughout: 5% edge budget, 50 unrolled inner steps, GCN victim, five paired victim seeds, singleton-constrained. Columns are accuracy drops for uniform block sampling and for same-class and cross-class pools; each p is a paired t-test of that pool against uniform sampling, with everything else held fixed. **Bold** marks the pool CLASP's rule selects (cross-class when $h > 0.5$, same-class otherwise).

					same-class		cross-class	
Dataset	h	n	Clean	Uniform	Drop	p	Drop	p
Cora	0.810	2,708	80.14	14.60	12.14	0.0408	**17.56**	<0.0001
CiteSeer	0.736	3,327	68.84	19.24	7.32	0.0001	**19.46**	0.7642
PubMed†	0.802	19,717	78.24	61.04	9.38	<0.0001	**62.32**	0.6154
chameleon	0.230	2,277	65.88	8.29	**7.94**	0.5942	8.73	0.5210
squirrel	0.222	5,201	48.66	20.02	**16.23**	0.0037	18.89	0.0402
actor	0.217	7,600	30.78	2.25	**0.75**	0.0065	1.86	0.2478

†**PubMed.** A 62.32 pp drop takes accuracy to 15.9%, far below the 33.3% expected of random guessing on three classes. A degree audit (App. C) shows this is not hub construction, but that the 60 labelled nodes, 0.30% of the graph, absorb 12.5% of inserted edge endpoints. Under PubMed's label scarcity, corrupting labelled neighbourhoods appears disproportionately damaging. The GCN's confusion matrix on the poisoned graph shows systematic relabelling rather than noise: 69% of class-0 test nodes are predicted as class 2 and 48% of class-2 nodes as class 0, while a graph-free MLP on the same features still scores 72.8%. The poisoned neighbourhoods of the 60 labelled nodes teach the GCN a near-permutation of the classes, which is how its accuracy falls below chance. That is a different mechanism from the one this paper studies, and we do not count it as evidence for blinding.

Table 25: Datasets, referenced from Sec. 4. d is the feature dimension and h edge homophily, the fraction of edges joining same-class nodes; *labelled* is the share of training nodes. Budgets are 5% of undirected edges. Planetoid graphs use the public split; heterophilous graphs use the first geom-GCN split. PubMed's features are TF-IDF weights, binarized where the Jaccard filter or the feature channel is involved.

Type	Dataset	n	$\lvert E\rvert$	d	h	Classes	Labelled	Budget
Homophilous	Cora	2,708	5,278	1,433	0.810	7	5.2%	263
	CiteSeer	3,327	4,552	3,703	0.736	6	3.6%	227
	PubMed	19,717	44,324	500	0.802	3	0.3%	2,216
Heterophilous	chameleon	2,277	31,371	2,325	0.230	5	48.0%	1,568
	squirrel	5,201	198,353	2,089	0.222	5	48.0%	9,917
	actor	7,600	26,659	932	0.217	5	48.0%	1,332

Table 26: Threat-model audit and matched-total-budget control on all six graphs, under the protocol of Table 5: 5% edge budget, plain-GCN victim, five seeds. "New singletons" counts nodes the attack isolates, net of any it reconnects; negative means it reconnects previously isolated nodes. The matched-total row spends half the edge budget on edges and the other half on feature-bit flips.

Dataset	Setting	Drop (pp)	Edges changed	New singletons
Cora	CLASP, constrained (as reported)	17.56	260	0
	CLASP, constraint off	17.56	260	0
	Matched total budget	12.80	130	0
CiteSeer	CLASP, constrained (as reported)	19.46	225	−10
	CLASP, constraint off	19.46	225	−10
	Matched total budget	16.82	110	−7
PubMed	CLASP, constrained (as reported)	54.28	2,215	0
	CLASP, constraint off	54.88	2,215	5
	Matched total budget	49.36	1,105	0
actor	CLASP, constrained (as reported)	0.75	1,330	0
	CLASP, constraint off	0.51	1,330	8
	Matched total budget	0.71	665	0
chameleon	CLASP, constrained (as reported)	7.94	1,565	0
	CLASP, constraint off	8.11	1,565	0
	Matched total budget	4.61	780	0
squirrel	CLASP, constrained (as reported)	16.23	9,915	0
	CLASP, constraint off	16.02	9,915	5
	Matched total budget	11.26	4,955	0

Table 27: SimP-GCN (WSDM 2021; DeepRobust's implementation), a *learned* similarity-preserving defense, on CLASP's stored graphs at a 5% edge budget, three victim seeds. SimP-GCN builds a feature k-nearest-neighbour graph and self-supervised similarity targets from the features it is given; *clean kNN* builds both from the clean features instead, while the model still trains on the poisoned ones. The undefended GCN's drop on the same graph is shown for comparison. PubMed is not run: SimP-GCN materialises a dense $n \times n$ similarity matrix and its argsort, about 6 GB there.

			SimP-GCN drop (pp)		
Dataset	Graph	SimP-GCN acc.	as deployed	clean kNN	GCN drop (pp)
Cora	Clean	80.23	n/a	n/a	n/a
	Structure only	71.53	8.70	n/a	17.57
	Joint 5% X_{nz}	36.93	43.30	17.53	34.10
	Joint 10% X_{nz}	30.27	49.97	23.63	40.90
CiteSeer	Clean	70.73	n/a	n/a	n/a
	Structure only	64.00	6.73	n/a	19.10
	Joint 5% X_{nz}	33.30	37.43	13.57	40.97
	Joint 10% X_{nz}	22.93	47.80	15.43	46.50
actor	Clean	32.59	n/a	n/a	n/a
	Structure only	31.60	0.99	n/a	0.72
	Joint 5% X_{nz}	29.06	3.53	2.17	2.37
	Joint 10% X_{nz}	26.62	5.96	2.87	2.46
chameleon	Clean	50.44	n/a	n/a	n/a
	Structure only	48.46	1.97	n/a	7.75
	Joint 5% X_{nz}	48.76	1.68	2.27	5.77
	Joint 10% X_{nz}	53.44	−3.00	−0.22	5.70
squirrel	Clean	38.30	n/a	n/a	n/a
	Structure only	30.58	7.72	n/a	16.46
	Joint 5% X_{nz}	32.15	6.15	7.43	16.55
	Joint 10% X_{nz}	31.12	7.17	8.45	17.42

Table 28: Pro-GNN stress test on Cora, 5% edge budget, public split, DeepRobust's implementation with 200 epochs. Its alternating optimisation takes an SVD of the dense adjacency every epoch and is by far the most expensive defense here: one fit per poisoned graph took 41 to 160 minutes, so these are single values rather than repeated-seed estimates, at 200 epochs rather than the 1000 of DeepRobust's reference configuration, and on Cora only. That may understate Pro-GNN: against Metattack it ends below an undefended GCN (62.5% against 66.9%), so treat the absolute values as a lower bound. Metattack's graph is the one generated in Table 6.

Dataset	Graph	Pro-GNN accuracy (%)	Drop (pp)
Cora	Clean	80.90	n/a
	Metattack	62.50	18.40
	CLASP structure	66.30	14.60
	CLASP joint 5%	**46.90**	**34.00**

E THE 10/10/80 PROTOCOL

Table 21 repeats the edge-level measurement and the camouflage baseline in DeepRobust's largest-connected-component setting with a random 10/10/80 split, the protocol most of the Metattack and Pro-GNN literature uses, at the same 5% edge budget and with that harness's own victims. With 10% of nodes labelled the drops are smaller than under the public split, and the ordering of the two effects is the same: the filter keeps more of the attack's edges as its feature budget grows (65 to 95% on Cora, against 64% given the clean features), the defended model ends below the undefended one at a 10% budget (30.0 against 27.7 pp), and the camouflage copies take PRBCD's inserted edges from 68% kept to all of them for 1.1 pp more loss. CiteSeer behaves the same way.

F METACON, RUN IN ITS AUTHORS' HARNESS

Metacon is the closest published competitor with public code. We run it from its authors' code in their own setup (largest connected component, random $10/10/80$ split, their surrogate and hyperparameters), with Metattack and CLASP on the same graph, split and budget, which its memory limits to 1% and 20 inner steps (App. T). There the structure-only attacks are all weak, 0.7 to 2.2 pp, and none leads on both graphs, while the joint attack costs 7.73 and 7.63 pp and GCN-Jaccard loses more than an undefended model (10.30 against 7.73 on Cora) although it removes *more* edges ($548 \rightarrow 756$; Table 14). Pruning *more* edges while the defended model loses *more* accuracy is the opposite sign to the public split, where the count falls ($572 \rightarrow 547$ at a 5% feature budget); the count mixes inserted and clean edges, so neither direction measures blinding, which is why Sec. 5.2 counts inserted edges instead. This protocol has 1.8 times the public split's training nodes, and App. E runs the edge-level measurement inside it at the paper's own 5% budget, where both findings reproduce.

G A PURIFIER THAT LEARNS ITS EVIDENCE

The defenses broken above all read the raw features. The standing objection is that a defense which learns a representation first would not be fooled so cheaply. We test that with a STABLE-style purifier (simplified): a one-layer GCN encoder trained by Deep Graph Infomax (Veličković et al., 2019), then a cosine cut on the embeddings that removes the least similar tenth of the edges, then the paper's usual victim, which is the encode-then-prune design STABLE (Li et al., 2022) uses without its recovery step. The implementation is ours, not the authors' code, so this is evidence about that design and not about their release or their reported numbers.

Table 29 applies the same counterfactual as the rest of the paper: the embeddings are learned either from the features the attacker wrote or from the clean ones, while the victim always trains on the poisoned features. Against structure alone the two agree exactly, as they must, since nothing has touched the features. At a 10% feature budget they separate: on Cora the purifier keeps 85% of the attack's inserted edges as deployed and 62% from clean evidence, on CiteSeer 78% against 45%, and the gap is worth 5.7 and 2.1 points of accuracy. At *this* budget, learning the representation adds a layer in front of the evidence rather than moving it out of the attacker's reach, and the blinding it suffers on Cora is the same size as GCN-Jaccard's. That is a statement about a 10% feature budget only. App. H asks the cheap question instead, and there the answer reverses: the one word per edge that blinds GCN-Jaccard completely does not blind this purifier at all.

Table 29: A defense that *learns* the evidence it filters on, a STABLE-style purifier: a one-layer GCN encoder trained by Deep Graph Infomax, then a cosine cut on the embeddings that drops the least similar tenth of the edges, then the paper's usual victim. *As deployed* learns the embeddings from the features the attacker wrote; *clean evidence* learns them from the clean features while the victim still trains on the poisoned ones, which is the counterfactual of Table 1. *Kept* is the share of the attack's inserted edges that survives the purifier, and drops are in points against its own clean accuracy, three victim seeds. The structure-only row flips no features, so the two conditions must agree there; they do.

		inserted edges kept		drop (pp)	
Dataset	Flips (% X_{nz})	as deployed	clean ev.	as deployed	clean ev.
Cora (clean 79.53)	0	69%	69%	15.33	15.33
	0.25	68%	69%	20.37	20.30
	10	85%	62%	45.10	39.37
CiteSeer (clean 67.50)	0	53%	53%	13.23	13.23
	0.25	55%	52%	19.47	18.93
	10	78%	45%	46.67	44.57

H DO THE COPIES BLIND A DEFENSE THAT LEARNS ITS EVIDENCE?

Every defense the copies break in Sec. 5.3 reads the raw features. App. G shows that a purifier which learns its evidence is blinded by a 10% joint feature budget, but that budget rewrites more than half of Cora's test nodes. The question this appendix answers is the cheap one: does the *minimal* forgery,

the one word per inserted edge that Prop. 1 prices, also blind a defense that learns a representation before it filters?

The attacks are Metattack and PRBCD, which never see a defense and never edit a feature, so the only feature edits are the copies themselves. Five feature conditions per stored graph on Cora and CiteSeer: none, Prop. 1's minimal copies, and the check-aware copies of App. N at one, two and three per inserted edge; on the other four graphs the three that bracket them, because one cell costs twenty to forty minutes here. Two defenses: the STABLE-style purifier of App. G and SimP-GCN, each run in both evidence conditions with three victim seeds. PubMed is measured on the purifier alone, since SimP-GCN's dense $n \times n$ similarity matrix does not fit at its n, and at $\tau = 0.1$, the threshold at which a Jaccard filter is active there. Table 30 gives all 35 cells.

The answer is no, and the margin is not close. On Cora the 136 copies that take GCN-Jaccard from keeping 124 of Metattack's 263 inserted edges to keeping all of them take the purifier from 161 to 164; three copies per edge, at 785 words, reach 187. On CiteSeer 57 copies take GCN-Jaccard from 169 of 227 to all of them and leave the purifier exactly where it was, at 125. The minimal copies make GCN-Jaccard keep every inserted edge on five of the six graphs and all but 20 of PubMed's 2,214, while the largest they move the purifier is 63 of chameleon's 1,304. In accuracy the largest gap between the two evidence conditions, over 26 cells that flip a feature, is 2.8 points for the purifier and 3.5 for SimP-GCN, and the sign is not even stable: on 14 of those 26 cells the purifier does *better* with the attacker's evidence than with the clean evidence. The one place the copies move many edges is squirrel, where two check-aware copies per edge take the purifier from 5,635 of 8,600 to 7,263; even there the defended accuracy is 1.2 points *higher* with the attacker's evidence than without it, so the extra edges cost the defender nothing.

How cheap the copies are is not the same question on every graph, and the $\%\, X_{nz}$ column of Table 30 is the one to read. On Cora and CiteSeer the minimal copies are 0.05 to 0.28% of the non-zero feature entries and two check-aware copies per edge stay at or below 1.07%: cheap by the unit the rest of the paper uses. On the heterophilous graphs they are not. Two copies per edge cost 8.6% of chameleon's entries and 18.0% of squirrel's, which is a larger edit than the 10% joint budget that blinds these defenses on Cora. So the claim that a *cheap* forgery does not buy a learned defense rests on Cora, CiteSeer and PubMed, where the copies really are cheap; on chameleon, squirrel and actor the table shows only that copies of that size do not buy one either.

Two things bound how much this can be read as a defense succeeding. First, these defenses were not removing the attack's edges to begin with: given clean evidence the purifier keeps 161 of Metattack's 263 inserted edges on Cora and 211 of PRBCD's, because it prunes only the least similar tenth of all edges, so there is less evidence to corrupt than GCN-Jaccard offers. App. J tightens that cut until the purifier removes almost every inserted edge and finds the same answer. Second, SimP-GCN is not exactly reproducible across runs on this machine: the rows that flip no feature, where the two conditions are the same computation, differ by up to 0.9 points, so its gaps below that are not measurements of anything. The purifier is deterministic and its no-copy rows agree exactly.

What separates this from the 10% joint budget is at least the size of the edit, and it is not only the size: App. I spends about the same number of words as the check-aware copies here, aims them at the embedding instead, and gets a positive gap in all four cases where these copies do not. The copies change one word per edge and leave what a node says intact; a 10% budget rewrites more than half of Cora's test nodes, and a clean-trained MLP changes its prediction on about 30% of them. A defense that learns its representation from the features is blinded by the second and not by the first, which puts a floor under the attacker's price that the raw-feature rules do not have. We read that as the one encouraging result in this paper for the defender, and as the direction a defense should take, not as evidence that learned purification is safe: the budget that defeats it is one a joint attack already reaches, and on PubMed the check-aware copies are built for the corroboration check rather than for $\tau = 0.1$, so their Jaccard column there is lower than the minimal copies' by construction.

Table 30: Does the cheap forgery blind a defense that *learns* its evidence? The attacks are Metattack and PRBCD, which never see a defense and never edit a feature; the only feature edits are the copies that pay the filter's price. *min* is Prop. 1's minimal camouflage; $+m$ are the check-aware copies of App. N, m per inserted edge. The defenses are the STABLE-style purifier of App. G and SimP-GCN. Each is run twice: *dep.* gives it the features the attacker wrote, *cln.* gives it the clean ones while the victim still trains on the poisoned ones. *Kept* is the share of inserted edges the defense's filter leaves; GCN-Jaccard is shown for comparison. % X_{nz} is what the copies cost as a share of the feature matrix's non-zero entries, which is the unit used everywhere else in the paper: on Cora and CiteSeer the copies are cheap by that measure and on the heterophilous graphs they are not, reaching 18% on squirrel, so only the first two graphs test a *cheap* forgery against these defenses. Drops are in points against each defense's own clean accuracy, three victim seeds. The *none* row flips no features, so its two conditions must agree.

		words bought		inserted edges kept			purifier drop		SimP-GCN drop	
Dataset	Copies	count	% X_{nz}	Jaccard	pur. dep.	pur. cln.	dep.	cln.	dep.	cln.
Metattack on Cora										
	none	0	0.00	47%	61%	61%	15.23	15.23	14.60	14.60
	min.	136	0.28	100%	62%	61%	14.37	14.90	13.47	13.77
	+1	263	0.53	100%	63%	61%	14.90	15.23	14.30	13.87
	+2	524	1.06	100%	67%	61%	15.17	15.03	14.20	13.97
	+3	785	1.60	100%	71%	61%	15.97	16.10	14.97	14.43
PRBCD on Cora										
	none	0	0.00	67%	80%	80%	13.80	13.80	9.93	9.93
	min.	86	0.17	100%	83%	80%	14.13	13.97	10.47	10.63
	+1	263	0.53	100%	83%	80%	15.27	14.27	11.17	10.97
	+2	525	1.07	100%	83%	80%	15.37	14.07	12.07	10.40
	+3	785	1.60	100%	85%	80%	16.03	14.77	12.87	11.20
Metattack on CiteSeer										
	none	0	0.00	74%	55%	55%	11.90	11.90	6.60	6.60
	min.	57	0.05	100%	55%	55%	13.13	12.50	6.60	6.77
	+1	227	0.22	100%	58%	55%	12.90	12.90	7.33	7.03
	+2	454	0.43	100%	59%	55%	15.90	13.07	6.37	7.43
	+3	681	0.65	100%	63%	55%	14.27	12.00	7.20	7.40
PRBCD on CiteSeer										
	none	0	0.00	76%	76%	76%	14.27	14.27	12.37	11.47
	min.	55	0.05	100%	76%	76%	13.17	13.43	10.70	12.53
	+1	227	0.22	100%	77%	76%	13.77	13.90	12.67	12.80
	+2	454	0.43	100%	80%	76%	14.13	14.60	12.53	13.30
	+3	681	0.65	100%	81%	76%	14.47	14.93	13.07	12.27
Metattack on chameleon										
	none	0	0.00	15%	81%	81%	11.40	11.40	7.89	7.89
	min.	1,248	4.28	100%	83%	81%	10.53	11.99	7.97	7.31
	+2	2,947	10.11	100%	84%	81%	11.92	11.92	10.16	7.68
PRBCD on chameleon										
	none	0	0.00	11%	67%	67%	40.57	40.57	26.54	26.54
	min.	1,093	3.75	100%	72%	67%	43.06	41.59	30.70	30.04
	+2	2,498	8.57	100%	74%	67%	42.25	41.96	33.63	30.12
PRBCD on squirrel										
	none	0	0.00	34%	66%	66%	45.85	45.85	31.70	31.70
	min.	5,460	5.84	100%	72%	66%	43.93	45.79	32.72	33.53
	+2	16,858	18.03	100%	84%	66%	45.50	46.65	34.42	33.75
PRBCD on actor										
	none	0	0.00	57%	98%	98%	14.58	14.58	3.82	3.82
	min.	536	1.31	100%	99%	98%	14.25	16.01	4.85	4.69
	+2	2,517	6.14	100%	99%	98%	16.45	16.56	6.49	4.89
PRBCD on PubMed										
	none	0	0.00	42%	82%	82%	54.00	54.00	n/a	n/a
	min.	4,546	0.46	99%	83%	82%	53.50	53.37	n/a	n/a
	+2	4,428	0.45	68%	82%	82%	54.00	54.33	n/a	n/a

I AN ATTACKER WHO KNOWS THE PURIFIER

App. H has the same hole this paper criticises elsewhere. The copies it uses were bought to pass a Jaccard threshold, so they were never aimed at a defense that learns its evidence, and a counter-measure has to be tested against an attacker who knows it (Mujkanovic et al., 2022). This appendix builds that attacker and repeats the measurement.

The purifier is handed over in full: architecture, training procedure, seed and keep fraction. The attacker then buys words that raise the *embedding* cosine of its own inserted edges rather than their feature overlap. For each inserted edge below the cut it scores every word the donor has and the receiver lacks by how much adding it would raise $\cos(z_{\text{receiver}}, z_{\text{donor}})$, buys the best one, and re-trains the encoder; five rounds, at most five words per edge. The score is exact for the encoder's linear part, since a one-layer GCNConv computes $h = \hat{A}XW$ before its non-linearity, so writing word j into node r moves h_i by $\hat{A}_{ir}W_j$. The defender then re-trains its own encoder from scratch and the 2×2 runs as before.

Aiming at the embedding is the better attack, and it is cheaper. Table 31: on Cora against Metattack it buys 420 words, 0.85% of the non-zero entries, and takes the purifier from keeping 161 of 263 inserted edges to 190, for a gap of 1.67 points. The check-aware copies of Table 30 spent 785 words, nearly twice as many, to reach 187 edges and -0.13 points. Across the four cases the gap is positive every time, 1.67 to 2.47 points, where the copies bought for the raw-feature rule ranged from -0.47 to 2.27 and averaged under a point. So the earlier result overstated the floor: an attacker who aims at the right target does buy something, reliably, at a fraction of the price.

It does not remove the floor. The largest gap an embedding-aware attacker reaches here is 2.47 points, against the 5.7 that a 10% feature budget buys on the same purifier and the same graph, and against a raw-feature filter that is blinded *completely* for 136 words, a third of what this attack spends. The words it buys are visibly aimed elsewhere: they leave GCN-Jaccard keeping 200 to 210 of Cora's 263 inserted edges rather than all of them, because they were not bought for that threshold. The honest statement is therefore narrower than App. H's: per point of blinding, learning the evidence costs the attacker 4.6 to 14.5 times what blinding a Jaccard filter costs, and in raw words 2.3 to 7.7 times the 55 to 136 that blind a Jaccard filter *completely*, for partial blinding rather than complete. That is a real price rise and not a wall, and a stronger adaptive attack than ours, one that optimises the encoder's training rather than treating it as fixed between rounds, may well do better.

Table 31: An attacker who knows the purifier. The copies of Table 30 were bought to pass a Jaccard threshold, so they were never aimed at a defense that learns its evidence. Here the attacker is given the purifier's architecture, training procedure, seed and keep fraction, and buys the word that most raises the *embedding* cosine of each of its own inserted edges, re-training the encoder between rounds, up to five words per edge. The defender then re-trains its encoder from scratch. *Kept* is the share of inserted edges the purifier leaves; *as dep.* learns the embeddings from the features the attacker wrote and *clean ev.* from the clean ones, while the victim always trains on the written ones. *Gap* is how much more the defended model loses as deployed, in points; a positive gap is blinding. The last column is what GCN-Jaccard does with these same words, which were not bought for it. Three victim seeds.

		words bought		inserted edges kept		purifier drop		
Dataset	Attack	count	% X_{nz}	as dep.	clean ev.	as dep.	gap	Jaccard
Cora	Metattack	420	0.85	72%	61%	16.67	1.67	80%
	PRBCD	197	0.40	89%	80%	16.77	2.47	76%
CiteSeer	Metattack	437	0.42	72%	55%	14.80	1.77	92%
	PRBCD	235	0.22	82%	76%	15.30	1.70	85%

J THE PURIFIER'S CUT, SWEPT

App. H leaves one objection open. The purifier there drops the least similar tenth of the edges, so it keeps most of an attack's inserted edges even when its embeddings come from the clean features,

and a defense that barely filters cannot be blinded much. If that is the whole explanation, the gap between the two evidence conditions should open as soon as the cut is tight enough to matter.

Table 32 sweeps the cut from a tenth to a half of the edges against the minimal copies. It does what a stricter cut should: on Cora against Metattack the purifier goes from keeping 164 of 263 inserted edges to keeping 11, at a cost of 4.4 points of clean accuracy. The gap does grow, from -0.5 points at the loosest cut to 1.5 at 0.6 and 1.4 at 0.5, so the loose cut was hiding a little blinding. But 1.5 points is the largest value anywhere in the sweep on Cora, and against PRBCD the gap never leaves ± 0.4; on CiteSeer it reaches 2.2 once and is negative in four of eight cells. A cut strict enough to remove 96% of Metattack's inserted edges still gives the copies about a point and a half, where the same copies take GCN-Jaccard from removing 53% of them to removing none.

So the finding of App. H is not an artefact of a weak cut. It does come with a price the defender pays in clean accuracy: at the cut where the purifier removes almost every inserted edge it has given up 4.4 points on the clean graph, which is the same trade Table 15 shows for a Jaccard threshold, and the reason nobody deploys the strict setting.

Table 32: Is the purifier only unblinded because it filters so little? The cut is swept from the least similar tenth of the edges, the setting used everywhere else, down to half of them, on the graphs Prop. 1's minimal copies camouflage. *Kept* is the share of the attack's inserted edges that survives the purifier; *as dep.* learns the embeddings from the features the attacker wrote and *clean ev.* from the clean ones, while the victim always trains on the copied-to features. *Gap* is how much more the defended model loses as deployed than with clean evidence, in points; a positive gap is blinding. Accuracies are against each cut's own clean accuracy, three victim seeds.

			inserted edges kept		defended accuracy	
Dataset	Attack	keep	as dep.	clean ev.	as dep. → clean ev.	gap
Cora	Metattack	0.9	62%	61%	65.17 → 64.63	−0.53
		0.75	27%	26%	66.03 → 66.90	0.87
		0.6	10%	7%	68.27 → 69.80	1.53
		0.5	4%	4%	69.37 → 70.73	1.37
	PRBCD	0.9	83%	80%	65.40 → 65.57	0.17
		0.75	55%	52%	66.27 → 66.63	0.37
		0.6	32%	33%	66.93 → 67.10	0.17
		0.5	24%	24%	68.33 → 68.17	−0.17
CiteSeer	Metattack	0.9	51%	51%	56.30 → 56.87	0.57
		0.75	27%	26%	61.20 → 63.37	2.17
		0.6	13%	13%	64.03 → 63.70	−0.33
		0.5	9%	9%	62.93 → 63.57	0.63
	PRBCD	0.9	74%	76%	54.83 → 54.63	−0.20
		0.75	47%	47%	57.73 → 58.57	0.83
		0.6	25%	24%	59.20 → 59.17	−0.03
		0.5	17%	16%	61.00 → 60.00	−1.00

K WHICH SIDE OF NORMAL

App. N reads its scores in one direction and calls an AUC below 0.5 a failure. It is not one: an AUC as far below chance separates two classes exactly as well as the same distance above it, with the sign reversed. What the number says is that the attack's edges are not average-looking. They are unusually *well* corroborated, which is what an optimizer pushed to make an edge survive a similarity rule should produce, and it is the opposite of what the minimal copies of Table 2 produce, which is edges that share the fewest words that will pass.

Table 33 and Fig. 4 follow that across the feature budget. The AUC of flagging the attack's edges as the least corroborated falls monotonically on both graphs, from 0.677 at a zero feature budget to 0.320 at 10% on Cora and from 0.764 to 0.364 on CiteSeer, crossing chance between 1% and 2%. A defender pruning the least corroborated tenth of the surviving edges gains 1.8 pp on Cora against structure alone and loses 2.9 at 10%; pruning the most corroborated tenth loses 1.1 and gains 0.5.

The accuracy differences are small, within about 3 pp over three victim seeds, and not uniformly signed; the robust part of this result is the ranking, which moves by 0.357 of AUC or more in the same direction on both graphs.

The practical reading is that a one-sided similarity test has a sign, the attacker chooses which side to sit on by choosing a budget, and the defender sets the sign in advance without seeing it. It also explains why the check-aware copies of Table 35 are the hardest case: two corroborated copies per edge put the attack near chance in *both* directions, which is the one place neither tail reaches.

Table 33: Which side of normal the attack's edges sit on. Among the edges GCN-Jaccard keeps, each edge is scored by how far the words its endpoints share are corroborated by either endpoint's other neighbours. *AUC least* is the ROC-AUC of flagging the attack's inserted edges as the *least* corroborated, and *AUC most* of flagging them as the *most*; the two sum to one, and whichever exceeds 0.5 is the tail the attack occupies. Δ is what dropping the 10% most suspicious surviving edges by that rule does to the defended accuracy, over three victim seeds. The defender must choose a direction without observing the attacker's feature budget.

		AUC		Δ accuracy (pp)	
Dataset	Flips (% X_{nz})	least	most	prune least	prune most
Cora	0	0.677	0.323	1.80	−1.10
	0.25	0.578	0.422	0.53	−0.17
	1	0.493	0.507	−2.23	0.60
	2	0.416	0.584	−1.87	1.40
	5	0.334	0.666	−3.30	1.30
	10	0.320	0.680	−2.87	0.47
CiteSeer	0	0.764	0.236	1.33	−0.57
	0.25	0.679	0.321	0.93	−0.50
	1	0.542	0.458	−0.70	0.13
	2	0.487	0.513	−0.67	0.77
	5	0.383	0.617	−1.70	0.80
	10	0.364	0.636	−0.70	−0.60

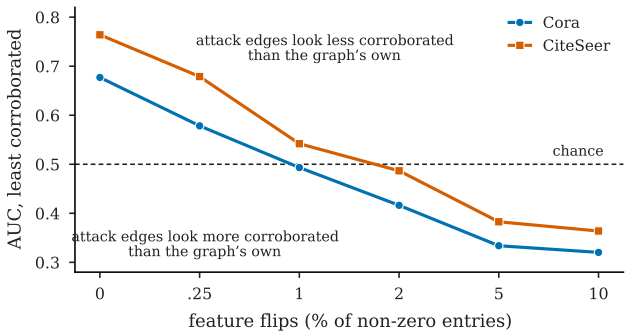

Figure 4: The attack crosses from one side of normal to the other. Among the edges GCN-Jaccard keeps, the ROC-AUC of flagging the attack's inserted edges as the *least* corroborated by their endpoints' other neighbours, against the feature budget. Above chance the attack undershoots the corroboration of a real edge and a defender should prune the least corroborated; below it the attack overshoots and that rule helps the attacker instead.

L Multiple comparisons

The paper reports several dozen hypothesis tests, and each table asks its tests together, so each is a family. We apply Holm's step-down correction within the two families large enough for the distinction to matter, and report what it costs. Holm is used rather than Bonferroni because it is uniformly more powerful and assumes nothing about independence.

The joint-versus-sequential comparison (Table 17) is 24 paired t-tests; 21 reach 0.05 raw and 16 after correction, the five lost being on PubMed, actor and squirrel. The claim the section makes from that

table is a count of wins, 17 of 24, which does not depend on significance at all. The candidate-pool sweep (Table 24) is 12 tests; 7 reach 0.05 raw and 4 after correction. The pool CLASP's rule selects still beats uniform sampling on Cora (adjusted $p < 10^{-4}$) and still loses to it on squirrel with the same-class pool (adjusted $p = 0.033$), but the cross-class pool's loss on squirrel (raw $p = 0.040$) does not survive, and neither does the same-class pool's on actor (adjusted $p = 0.052$). Section 5.4 states the corrected versions.

Every other p in the paper compares one attack against CLASP on one graph over shared victim seeds. Those are reported raw and read as descriptive: they say how large a difference is relative to victim-training noise on that graph, not whether a family of hypotheses survives. `multiple_comparisons.py` ships with our artifacts and recomputes all of the above.

M CONTINUOUS FEATURES

Every other result in the paper is on sparse binary attributes, where the Jaccard rule at $\tau = 0.01$ reduces to sharing one word and the forgery is a copied word. Neither statement has an analogue on dense continuous attributes, which most current benchmarks carry. PubMed answers part of it without leaving the paper's data: its features are TF-IDF weights, binarized everywhere else here, and the natural filter on them is cosine similarity, which is also what GNNGuard uses. They are still bag-of-words counts, so what this tests is whether the effect needs *binary* attributes, not whether it survives learned embeddings, which we do not test.

Table 34 repeats the camouflage construction in that setting. The continuous counterpart of copying a word is moving the lower-degree endpoint of an inserted edge a step towards the other until the pair passes, touching nothing else. It works at every threshold, and the trade is the one Table 15 shows for binary features: at $\tau = 0.05$ the filter costs the defender nothing on the clean graph and removes half of PRBCD's inserted edges, and 0.58% of the feature matrix's mass restores all of them and takes the defended accuracy from 49.9 to 27.8%; at $\tau = 0.2$ the filter removes all but 3.5% of them, at 40.1% of the clean edges and 4.0 points of clean accuracy, and the attacker's price rises to 2.87% of the mass. The mechanism is not an artefact of binarization.

That construction is feasible but not minimal, so every price it reports is an upper bound, and Prop. 2 gives the floor underneath it. Priced edge by edge on the same graphs, the median edge the filter rejects costs 0.0038 of its moving endpoint's norm at $\tau = 0.01$, rising to 0.1533 at $\tau = 0.2$, and summed over the rejected edges the exact minimum is 0.004% of the matrix's ℓ_2 mass at $\tau = 0.01$ against the interpolation's 0.060%: the construction overpays by 16.7 times there, and by 1.4 times at $\tau = 0.2$, where almost every edge must move. The conclusion the table draws is therefore conservative in the attacker's disfavour, and most so at the threshold a defender would actually deploy.

The relative mass moved is only meaningful beside a noticeability check, since the move is aimed at another node rather than spread at random. At $\tau = 0.05$ the construction touches 1,046 nodes and changes the median touched node by 9.7% of its own vector; a graph-free MLP trained on the clean features changes its prediction on 1.2% of them, against 0.3% for a random move of the same size at the same nodes. At $\tau = 0.1$ those become 16.0% and 1.8% against 0.4%. The aimed move is thus four times more visible to a feature-only classifier than an aimless one of equal size, and still far below the 3 to 14% the binary copies reach (Sec. 7).

Table 34: The mechanism on *continuous* features. PubMed's TF-IDF weights are left as they are, not binarized as everywhere else in the paper, so Jaccard does not apply and the filter is cosine similarity, which is what GNNGuard uses. *Pruned* is the clean edges the filter removes and *cost* the clean accuracy it gives up to do so. For each stored structural attack: the share of its inserted edges the filter keeps, then the same after the continuous camouflage, which moves the lower-degree endpoint of every inserted edge a step towards the other until the pair passes and touches nothing else. *Moved* is the resulting change in the feature matrix as a share of its total mass, the continuous counterpart of the copied words of Table 2. That construction is feasible but not minimal, so *moved* is an upper bound; *price* is the exact minimum of Prop. 2, the median over the inserted edges the filter rejects of $\sin(\theta - \arccos\tau)$, the smallest change to one endpoint as a fraction of that endpoint's own norm. The last column is the defended accuracy itself, not a drop, before and after the camouflage, over three victim seeds.

	clean graph			inserted edges kept		attacker pays		defended acc.
τ	pruned	cost (pp)	Attack	before	after	moved	price	before $\rightarrow$ after
0.01	172 (0.4%)	−0.07	PRBCD	93.7%	100.0%	0.07%	0.0038	29.70 $\rightarrow$ 27.90
			CLASP	91.7%	100.0%	0.08%	0.0043	24.80 $\rightarrow$ 23.80
0.05	2,220 (5.0%)	−0.13	PRBCD	49.8%	100.0%	0.58%	0.0236	49.90 $\rightarrow$ 27.77
			CLASP	46.0%	100.0%	0.54%	0.0261	52.50 $\rightarrow$ 21.83
0.1	6,827 (15.4%)	0.43	PRBCD	20.5%	100.0%	1.30%	0.0607	64.73 $\rightarrow$ 29.40
			CLASP	17.1%	100.0%	1.20%	0.0639	69.33 $\rightarrow$ 23.00
0.2	17,764 (40.1%)	4.00	PRBCD	3.5%	100.0%	2.87%	0.1533	71.97 $\rightarrow$ 33.00
			CLASP	4.0%	100.0%	2.54%	0.1584	70.63 $\rightarrow$ 23.47

N AN EDGE-AWARE CHECK

The detectors of App. O score nodes, and a one-word change barely moves a node. The forgery's signature is on the *edge*: the pair passes the filter on words that nothing else about either endpoint corroborates. We score each edge the filter keeps by three quantities computed from the poisoned graph and features alone: how many words the endpoints share, how far those words are corroborated by each endpoint's other neighbours, and how rare they are in the corpus. Each is read in the direction the data indicates rather than a fixed one, because an AUC below chance separates the classes exactly as well as the same distance above it; the table reports $\max(A, 1 - A)$ and the tail the attack's edges occupy. No one of the three dominates, so all three are given and the strongest is quoted, which a defender could not choose in advance.

Table 35 gives all three scores and the pipeline they suggest. Against the minimal copies the check works: on Cora the best score reaches 0.835 and dropping the 10% most suspicious surviving edges removes 177 of 263 inserted edges, recovering the whole 8.0 points the copies cost and 2.5 points beyond it, and on CiteSeer it reaches 0.884.

It works because the minimal copies are minimal. An attacker told about the check copies more words per edge and picks them among the donor's words for the ones the receiver's other neighbours already hold. On Cora that takes Metattack's camouflage from 136 copies to 524, about one and a half extra words per inserted edge and 1.06% of the non-zero entries; the best score falls to 0.594 and the pruning removes 7 inserted edges instead of 177, costing the defender accuracy. The pattern holds on CiteSeer and for PRBCD's graph, where the pruning's benefit shrinks from 8.4 points to 0.3, though it does not vanish: against PRBCD on Cora it still returns about 1.5 points at two and three copies. The check is therefore a price rise of one or two words per edge, which is what the k-robust rule of Sec. 5.4 and a higher threshold also are.

What the check-aware copies buy is not invisibility but the middle: at two copies per edge the strongest score sits at 0.594 on Cora, near chance in either direction, which is the one place neither tail reaches. The signature moves rather than vanishing, and a defender who watched both tails would still see something: at three copies the rarity score rises to 0.676 while the shared-word count falls to 0.347, so those edges share unusually *many* rare words. By the same accounting that would cost the attacker another word per edge.

AUC and pruning can disagree, because a fixed pruning budget only sees the extreme tail: on CiteSeer at two copies the strongest score still reaches 0.670, yet the pruning removes 15 of the 227 inserted edges, fewer than the 23 a random tenth would take.

Reading the direction from the data matters most on chameleon. There the camouflaged edges are the *most* corroborated rather than the least, at 0.784 against Metattack's graph and 0.753 against PRBCD's, and pruning that tail removes 529 of PRBCD's $1{,}304$ inserted edges for 252 clean ones and lifts the defended accuracy from 15.5 to 24.1%. That recovery of 8.6 points is second in the paper only to the 10.5 the same rule recovers on Cora against the minimal copies, where the tail runs the other way. Scored in a fixed direction the same rule would have looked like the check's worst graph rather than its best, which is why the table reports the tail alongside the number.

Table 35: An edge-aware check, scored where a defender would use it: among the edges GCN-Jaccard keeps. All three scores are computed from the poisoned graph and features alone. *Corroboration* asks, for each word the endpoints share, what fraction of each endpoint's *other* neighbours also has it; *shared* counts the shared words; *rarity* takes the corpus rarity of the best shared word. Each is the ROC-AUC of separating the attack's inserted edges from the graph's own, and a value below 0.5 means the attack's edges look *better* than the graph's own by that score. The right block drops the 10% most suspicious surviving edges by the best score and retrains (three victim seeds), on the tail the best score points at. *Tail* names that tail: *least* when the attack's edges are the least corroborated, fewest shared or rarest by the best score, *most* when they are the opposite, which an AUC below 0.5 reports. A one-sided reading would have scored the *most* rows as failures. *$+m$ copies* are the check-aware copies: m per inserted edge, chosen among the donor's words for the ones the receiver's other neighbours already hold.

			AUC by score				after dropping the top 10%		
Dataset	Graph	copies	corrob.	shared	rarity	tail	inserted	before	after
Cora	CLASP, structure		0.677	0.791	0.247	least	66/156	67.63	68.77
	CLASP, joint 0.25%		0.578	0.717	0.243	most	57/201	57.27	59.10
	CLASP, joint 10%		0.320	0.430	0.290	most	14/233	36.30	35.13
	Metattack + min. copies	136	0.741	0.835	0.394	least	177/263	62.27	72.77
	$+1$, check-aware	263	0.644	0.713	0.442	least	106/263	62.17	65.17
	$+2$, check-aware	524	0.582	0.514	0.594	least	7/263	62.10	61.17
	$+3$, check-aware	785	0.539	0.347	0.676	least	18/263	61.30	62.00
	PRBCD + min. copies	86	0.719	0.799	0.331	least	103/263	63.00	66.97
	$+1$, check-aware	263	0.566	0.638	0.404	least	44/263	62.97	63.83
	$+2$, check-aware	525	0.534	0.446	0.553	most	8/263	63.03	63.53
	$+3$, check-aware	785	0.525	0.296	0.644	most	29/263	63.07	63.73
CiteSeer	CLASP, structure		0.764	0.869	0.147	least	80/166	54.03	58.13
	CLASP, joint 0.25%		0.679	0.831	0.145	most	76/186	46.03	49.13
	CLASP, joint 10%		0.364	0.366	0.209	most	35/222	23.17	21.80
	Metattack + min. copies	57	0.778	0.884	0.189	least	126/227	55.73	57.80
	$+1$, check-aware	227	0.643	0.787	0.244	least	58/227	55.73	55.07
	$+2$, check-aware	454	0.587	0.665	0.330	most	15/227	55.27	53.37
	$+3$, check-aware	681	0.561	0.551	0.377	most	6/227	55.17	52.33
	PRBCD + min. copies	55	0.793	0.870	0.194	least	119/227	52.10	60.50
	$+1$, check-aware	227	0.622	0.775	0.236	least	64/227	52.90	54.70
	$+2$, check-aware	454	0.580	0.658	0.324	most	26/227	52.63	54.50
	$+3$, check-aware	681	0.561	0.545	0.406	most	10/227	52.33	53.73
chameleon	CLASP, structure		0.567	0.429	0.532	most	40/250	48.54	50.00
	CLASP, joint 0.25%		0.559	0.437	0.518	most	37/256	48.17	49.12
	CLASP, joint 10%		0.608	0.596	0.454	least	93/895	57.24	54.90
	Metattack + min. copies	1248	0.216	0.606	0.338	most	706/1523	37.79	41.30
	PRBCD + min. copies	1093	0.247	0.624	0.368	most	529/1304	15.50	24.12

O CAN A DEFENDER SEE THE FLIPS?

Before asking which edges a filter keeps, a defender facing this threat model would look at the features and try to flag the nodes the attacker edited. We score five unsupervised detectors, each fitted on the poisoned matrix alone because that is what a defender has: distance to the ten nearest nodes in feature space, the error of a rank-32 linear reconstruction, the node's word count as a z-score, the summed rarity of its words, and Nettack's co-occurrence test. Table 36 reports the best of the five on each poisoned matrix.

On the two graphs where the attack costs the most accuracy the best of the five stays close to chance. On Cora it reaches 0.535 AUC against CLASP's flips at a 0.25% budget and flags edited nodes at 2.8% precision where guessing gets 2.7%; at a 10% budget, with more than half the nodes edited, it reaches 0.578. On CiteSeer it never passes 0.616, which it reaches at the same 10% budget at 62.1% precision against a 55.5% base rate: better than guessing, but not by enough to act on when more than half the nodes are edited. The camouflage copies are the most visible case, because copying a neighbour's rare words raises the rarity score: 0.679 on Cora at 18.2% precision against a 4.9% base rate, so of the 132 nodes it flags, 24 are real. Detection is easier on squirrel and actor, where the flips are a larger share of a sparser matrix (up to 0.80 on squirrel at a 1% budget and 0.635 on actor), while chameleon sits with CiteSeer at 0.616. We read this as evidence that the copies are cheap to hide as well as cheap to make, not as a claim that no detector could work: a purpose-built or supervised one is untested.

Table 36: Can a defender see the flips? Five unsupervised detectors, each fitted on the poisoned feature matrix alone because that is what a defender has: mean distance to the ten nearest nodes in feature space, the error of a rank-32 linear reconstruction, the node's word count as a z-score against the corpus, the summed rarity $\sum_j -\log f_j$ of its words, and Nettack's co-occurrence test. Each is scored on how well it separates the nodes the attack edited from the rest. The table gives the best of the five by ROC-AUC, that detector's precision at k where k is the true number of edited nodes, and the base rate k/n that guessing achieves. *Camouflage* rows are the copies of Table 2, on the structural attack's graph.

		nodes edited		best detector		precision at k	
Dataset	Poisoned matrix	count	%	AUC	which	detector	guessing
Cora	CLASP, 0.25%	72	2.7%	0.535	word count	2.8%	2.7%
	CLASP, 1%	230	8.5%	0.541	co-occurrence	11.3%	8.5%
	CLASP, 5%	939	34.7%	0.548	word count	39.5%	34.7%
	CLASP, 10%	1469	54.2%	0.578	word count	59.9%	54.2%
	camouflage on Metattack	132	4.9%	0.679	word rarity	18.2%	4.9%
	camouflage on PRBCD	83	3.1%	0.569	reconstruction	10.8%	3.1%
CiteSeer	CLASP, 0.25%	139	4.2%	0.519	reconstruction	5.0%	4.2%
	CLASP, 1%	342	10.3%	0.583	reconstruction	16.4%	10.3%
	CLASP, 5%	1047	31.5%	0.605	co-occurrence	42.3%	31.5%
	CLASP, 10%	1846	55.5%	0.616	co-occurrence	62.1%	55.5%
	camouflage on Metattack	57	1.7%	0.569	word rarity	0.0%	1.7%
	camouflage on PRBCD	55	1.7%	0.557	word rarity	1.8%	1.7%
chameleon	CLASP, 0.25%	19	0.8%	0.549	word count	5.3%	0.8%
	CLASP, 1%	54	2.4%	0.584	word count	1.9%	2.4%
	CLASP, 5%	814	35.7%	0.531	co-occurrence	36.9%	35.7%
	CLASP, 10%	1511	66.4%	0.616	co-occurrence	71.2%	66.4%
	camouflage on Metattack	617	27.1%	0.633	reconstruction	32.1%	27.1%
	camouflage on PRBCD	546	24.0%	0.596	reconstruction	26.6%	24.0%
actor	CLASP, 0.25%	76	1.0%	0.455	co-occurrence	0.0%	1.0%
	CLASP, 1%	271	3.6%	0.543	co-occurrence	1.1%	3.6%
	CLASP, 5%	1032	13.6%	0.635	kNN distance	26.6%	13.6%
	CLASP, 10%	1942	25.6%	0.632	kNN distance	38.0%	25.6%
	camouflage on PRBCD	484	6.4%	0.686	reconstruction	15.1%	6.4%
squirrel	CLASP, 0.25%	38	0.7%	0.710	reconstruction	0.0%	0.7%
	CLASP, 1%	102	2.0%	0.801	reconstruction	17.6%	2.0%
	CLASP, 5%	2620	50.4%	0.550	co-occurrence	53.5%	50.4%
	CLASP, 10%	3641	70.0%	0.578	co-occurrence	73.9%	70.0%
	camouflage on PRBCD	1674	32.2%	0.713	reconstruction	51.5%	32.2%

P CONTROLS ON THE MECHANISM

Four controls, each of which re-scores graphs the paper already stores rather than running a new attack. Together they ask whether the blinding is an artefact of the feature budget's size, of its placement, or of chance.

The nodes the attack never touched. At a 10% budget the attack edits the features of 52.6% of Cora's test nodes, so part of the loss is the classifier's own input being rewritten rather than the graph being poisoned. Splitting the test set by whether a node's own features changed separates the two (Table 37). Edited nodes do lose more (53.0 against 27.4 pp on Cora at 10%), but the nodes the attack never touched still lose 27.4 pp where the same poisoned edges with clean features cost them 6.3: most of the damage reaches nodes through the graph, not through their own features. At 0.25% the question barely arises, since only 1.4% of Cora's test nodes are edited at all and the untouched drop, 22.0 pp, is the whole effect.

Would any flips do? Holding the adjacency at CLASP's own poisoned graph and changing only the feature matrix (Table 38): the attack's flips take the kept share to 80.4% at 0.25% and 100% at

10% on Cora, while the same number of flips placed at random leaves it at 63.7% and 60.4%, which is where the clean features leave it (63.6% and 62.7%). Random flips on the very same nodes, with the same per-node counts, do no better (63.5% and 61.4%). They also do almost no damage: at 10% the undefended accuracy is 64.1% under matched random flips against 39.3% under the attack's. The budget is not what blinds the filter; the placement is.

Is the word overlap chance? The paper reports that 65% of the words the attack adds at 0.25% are words the inserted neighbour already has. Table 39 gives four baselines that hold the nodes and the number of added words fixed. On Cora the observed share is 62.3% against 24.5% for a random node of the neighbour's class and 13.2% for a random node, so the targeting is far above chance. At 10% it falls to 16.7% against the same 12.2% baseline: the aimed, cheap form of the attack is a small-budget phenomenon, and the large budget is closer to indiscriminate feature damage.

Is it the shared objective? Table 40 scores the sequential attacks edge by edge. On Cora at a 5% budget the joint attack's filter keeps 97.4% of its inserted edges, while structure-then-features keeps 82.7% and features-then-structure 82.5%; at 10%, 100% against 94.9% and 83.1%. Running the two channels in sequence blinds the filter substantially, which is the mechanism's point, but the shared objective blinds it more.

Table 37: How much of the loss is the classifier's own input being rewritten. At each feature budget the test nodes are split by whether the attack changed that node's feature vector, and each subset is scored against the clean model's accuracy on the *same* subset (three victim seeds). *Edges only* gives the poisoned adjacency with the clean features, so the last two columns are the same nodes under the same edges, with and without the feature flips. The budget-0 row touches no features and is the consistency check.

			accuracy drop (pp)			edges only
Dataset	Flips (% X_{nz})	test nodes edited	all	edited	untouched	untouched
Cora	0	0.0%	17.57	n/a	17.57	17.57
	0.25	1.4%	21.70	0.00	22.01	18.70
	1	6.5%	25.13	20.51	25.45	16.26
	5	31.2%	34.10	46.26	28.59	12.31
	10	52.6%	40.90	53.04	27.43	6.26
CiteSeer	0	0.0%	19.10	n/a	19.10	19.10
	0.25	1.6%	23.97	25.00	23.95	21.38
	1	7.3%	31.77	28.31	32.04	22.15
	5	29.8%	40.97	53.69	35.57	14.43
	10	56.2%	46.50	54.98	35.62	10.88

Table 38: Would any feature flips do? The adjacency is held at CLASP's own poisoned one and only the feature matrix changes: the attack's, the clean one, the same number of flips placed uniformly at random, and the same number placed on the same nodes with the same per-node counts (three random draws each, three victim seeds). *Kept* is the share of that graph's inserted edges GCN-Jaccard keeps.

		inserted edges kept		accuracy (%)	
Dataset	Feature matrix	flips	kept	undefended	GCN-Jaccard
Cora, 0.25% flips	CLASP's flips	120	80.4%	58.47	57.27
	random, same nodes		63.5%	61.34	64.59
	random, matched count		63.7%	61.29	64.73
	clean (no flips)		63.6%	61.73	65.07
Cora, 10% flips	CLASP's flips	4920	100.0%	39.27	36.30
	random, same nodes		61.4%	60.42	61.43
	random, matched count		60.4%	64.11	66.51
	clean (no flips)		62.7%	64.50	66.17
CiteSeer, 0.25% flips	CLASP's flips	260	84.5%	44.70	46.03
	random, same nodes		76.7%	47.09	52.04
	random, matched count		76.8%	47.38	51.62
	clean (no flips)		76.8%	47.27	52.17
CiteSeer, 10% flips	CLASP's flips	10515	100.0%	22.17	23.17
	random, same nodes		62.0%	52.41	59.82
	random, matched count		72.5%	51.74	57.32
	clean (no flips)		71.6%	51.27	57.63

Table 39: Is the word overlap chance? For every word the attack adds at an endpoint of one of its own inserted edges, whether the inserted neighbour already has that word (*observed*), against four baselines that keep the same nodes and the same number of added words and change only which word, or whose vocabulary it is compared against.

			chance baselines (%)			
Dataset	Flips (% X_{nz})	observed	uniform word	frequency	same-class node	any node
Cora	0.25	62.3%	0.0	5.7	24.5	13.2
	10	16.7%	1.9	6.0	12.2	11.9
CiteSeer	0.25	37.8%	0.0	4.5	21.6	8.1
	10	9.2%	0.8	4.8	7.5	7.6

Table 40: Blinding under the sequential attacks of Table 17, at the same budgets: if running structure and then features blinded the filter as much as the joint attack, the shared objective would not be what causes it. The feature-only rows insert no edges, so they have no kept share. Three victim seeds; the filter and the survival count are those of Table 1.

Dataset	Attack	inserted	kept	undefended	Jaccard
Cora	Structure only	254	61.4%	62.60	67.63
	Features only, 5%	n/a	n/a	62.97	57.73
	Structure, then features, 5%	254	82.7%	53.57	51.27
	Features, then structure, 5%	240	82.5%	44.30	42.13
	Joint, 5%	233	97.4%	46.07	42.77
	Features only, 10%	n/a	n/a	52.37	47.50
	Structure, then features, 10%	254	94.9%	49.10	45.00
	Features, then structure, 10%	255	83.1%	42.73	40.97
	Joint, 10%	233	100.0%	39.27	36.30
CiteSeer	Structure only	223	74.4%	49.57	54.03
	Features only, 5%	n/a	n/a	33.47	36.07
	Structure, then features, 5%	223	93.7%	39.37	40.90
	Features, then structure, 5%	222	95.5%	37.40	39.77
	Joint, 5%	219	99.5%	27.70	28.50
	Features only, 10%	n/a	n/a	29.33	30.90
	Structure, then features, 10%	223	97.8%	30.37	31.53
	Features, then structure, 10%	223	97.8%	27.80	30.33
	Joint, 10%	222	100.0%	22.17	23.17

Q THE STRUCTURE-ONLY COMPARISON IN FULL

Table 5 runs the structure-only attacks on all six graphs at a 5% edge budget and Table 6 adds PGD, MinMax, time and peak memory. On Cora and CiteSeer, CLASP's structural component and PRBCD are the two strongest, and CLASP leads in the table's single draws, but only part of that lead survives regenerating both attacks with ten attack seeds (Table 7): they are level on Cora (16.94 against 16.56 pp, $p = 0.62$), CLASP is ahead on CiteSeer (19.83 against 16.81, $p < 10^{-4}$), and CLASP varies more across attack seeds (2.24 against 0.67 pp of standard deviation on Cora). Five seeds on the other four graphs keep every ordering but PubMed's, where the two are level ($p = 0.15$) and the single draw should not be read as a ranking; on PubMed every gradient-based attack drives the GCN from 78.8% to below chance, so the ranking there says little. On the heterophilous graphs the ordering reverses and CLASP is far weaker than PRBCD, whichever pool it samples ($p < 10^{-8}$ on each; on chameleon 7.94 pp against 52.72). The dense-matrix attacks are weak there too, so what separates PRBCD is not CLASP's sampling alone. Metattack beats CLASP on chameleon (12.28 against 7.94 pp, $p = 0.004$) and on Cora over three seeds; it holds dense $n \times n$ matrices and its time grows about as $n^2 B$, which by the rule fitted on Cora puts actor at a day, squirrel at 3.4 days and PubMed at 11. CLASP is the fastest optimization-based attack on Cora (17.4 s against 100.3 for PRBCD and 2090 for Metattack) and the most memory-hungry, needing 2.6 and 4.4 times PRBCD's on Cora and CiteSeer because the unrolled inner loop retains every activation (Table 6).

Adding the feature channel takes Cora and CiteSeer to 41.06 and 47.80 pp at a 10% feature budget (Table 4, with two adapter victims and three defenses). GCN-SVD resists best; GCN-Jaccard, which helps against structure alone (10.46 against 17.56 pp on Cora), gives nothing against it (41.62), and Sec. 5.2 measures why.

Switching Metattack's singleton constraint off gives the same attack on Cora and CiteSeer and moves the drop by at most 0.6 pp on the other four graphs, never significantly ($p \geq 0.18$; Table 26), so no gain comes from a looser threat model.

R ABLATIONS IN FULL

The candidate pool. With everything else fixed (Table 24), the cross-class pool beats uniform block sampling on Cora by 2.96 pp ($p < 10^{-4}$ after correcting that table's twelve tests together)

and ties on the other five, and the same-class pool the rule picks on heterophilous graphs is below uniform sampling on all three, significantly on squirrel alone: class conditioning is not why CLASP works.

Joint versus sequential, and the schedule. At equal budgets the shared objective beats running structure and features in sequence in 17 of 24 comparisons (Table 17), six of the seven exceptions on actor and chameleon, where the joint attack is weaker than structure alone to begin with, and the seventh on Cora (-1.77 pp, $p = 0.0031$). A four-times finer schedule (24.48 against 17.85 pp on Cora) and a margin loss both strengthen the structural attack; we keep five blocks and cross-entropy, so the structural CLASP numbers here are conservative (App. B).

S Proof of Proposition 1

A flip at an endpoint does one of four things: it copies a word the other endpoint has, so i rises by one and U is unchanged; it deletes a word the other endpoint lacks, so U falls by one; it deletes a shared word, so i and U both fall by one, which lowers the ratio; or it adds a word neither has, so U rises. Only the first two can help, so write $k = c + d$ for c copies and d deletions. The edge passes iff $(i+c)/(U-d) \geq \tau$, that is $k \geq \tau U - i + d(1-\tau)$. Since $\tau < 1$ this is smallest at $d = 0$, giving $k^\star = \lceil \tau U - i \rceil = \lceil \tau U \rceil - i$ for integer i, attained by copying alone. □

The bound is tight in both directions, so the price of one edge is known exactly rather than bracketed. Across many edges the problem couples: one copied word can serve several inserted edges that share an endpoint, and a word added at a node enlarges the union of its other edges. The camouflage counts of Sec. 5.3 are therefore forgeries that work, an upper bound on the cheapest joint forgery rather than its exact minimum.

Proposition 2. The set $\{y : \cos \angle(y, x_v) \geq \tau\}$ is a convex circular cone with axis x_v and half-angle $\varphi = \arccos \tau$. A point at angle $\theta \in (\varphi, \varphi + \pi/2)$ from the axis lies at distance $\|x_u\|_2 \sin(\theta - \varphi)$ from that cone, attained in the plane spanned by the point and the axis. Non-negativity of x_u and x_v gives $\theta \leq \pi/2$, and $\tau < 1$ gives $\varphi > 0$, so $\theta - \varphi < \pi/2$ and the point is not in the polar cone. The minimiser lies on the ray at angle φ from x_v, in the plane of x_u and x_v and between them, so it is a non-negative combination of two non-negative vectors and the unconstrained minimiser is already feasible. For $\theta = \pi/2$, $\sin(\pi/2 - \arccos \tau) = \cos(\arccos \tau) = \tau$. □

Proposition 1 prices one edge by flips at both endpoints; Proposition 2 prices one edge by moving one endpoint, so a node that is the moving endpoint of several inserted edges may pay more than the sum of the per-edge prices suggests. On the graphs of App. M that case is rare: 88 to 100% of the moving nodes carry exactly one inserted edge.

T Provenance, and what was not run

Metacon's memory. At its authors' 100 inner steps and a 1% budget, Metacon$_D$ reached 5.4 GB by its 25th of 50 perturbations, growing about 160 MB per perturbation, and was stopped; at 20 steps it grows about 45 MB per perturbation, which rules out a 5% budget. The authors' S configuration for Cora (MetaconGraD$_S$) also holds two dense $n \times n$ tensors per inner step; it reached 7.1 GB within two perturbations at 20 steps and ran out of memory at 30. Our copy of Metacon carries four memory patches (checkpointing of the contrastive block, a per-perturbation garbage collection and `malloc_trim`, and detaching stored tensors); each reproduces the released code's perturbed edges and adjacency checksum exactly.

SimP-GCN's cache. DeepRobust's SimP-GCN stores its feature kNN graph and its self-supervised targets on disk under a name that encodes only the feature matrix's shape, and loads them whenever a matrix of that shape comes back. A poisoned feature matrix has the clean matrix's shape, so scoring several graphs in one working directory silently reuses the evidence of the first. We give every graph its own directory. The clean-kNN column of Table 27 is that reuse made deliberate: the clean graph's cache with the poisoned graph's features.

Table 41: What was measured, and what was not.

State	Items
Measured, on all six graphs	The attack and defense grid (Table 4); every structural attack that fits, with time, isolated-process peak memory and paired tests (Tables 5 and 6); attack-seed replication of the structural comparison (Table 7); the edge-level analysis at every feature budget (Tables 1, 8 and 13); camouflage (Tables 2 and 9); threshold and cosine sweeps (Tables 15 and 16); the candidate-pool sweep (Table 24); decomposition and joint versus sequential (Table 17); budget response (Table 18); the threat-model audit and matched total budget (Table 26); and a median-aggregation GCN (Table 22).
Measured, on fewer	Blinding on the five binary-feature graphs (Table 10); SimP-GCN on five (Table 27); GNNGuard on five (Table 12); the learned purifier and the edge-aware check on two and three (Tables 29 and 35); the k-robust rule on three (Table 19); Metacon$_D$ in its authors' harness, attack seeds for the edge-level measurement, the geom-GCN split and three design alternatives on two (Tables 14, 23 and 11, App. B); Pro-GNN (Table 28) and the degree audit (App. C) on one. The four controls on the mechanism (App. P) run on Cora and CiteSeer, the two graphs where the attack's edges do the most damage.
Measured, with a stated deviation	Metacon$_D$ at a 1% budget and 20 inner steps rather than 5% and 100, because its memory grows by 45 to 160 MB per perturbation (see above); Metattack is also run at 20 steps as its matched comparator. Pro-GNN at 200 rather than 1000 epochs. PubMed's features binarized for the Jaccard filter and the feature channel.
Not run	Metacon$_S$ (memory, App. F); Metattack on PubMed, actor and squirrel, and PGD and MinMax on PubMed (dense $n \times n$ matrices and, for Metattack, 1 to 11 days by its measured scaling); GNNGuard and SimP-GCN on PubMed, and GCN-SVD and RGCN there, all of which build dense $n \times n$ matrices on that graph (the last two were killed by the out-of-memory killer, Table 4); Pro-GNN beyond Cora; a learned purifier run from its authors' code, rather than the reimplementation of App. G; ogbn-arxiv and any graph with learned embeddings; label rates other than the two splits; Meta-PGD; and EvA, for which we found no code release.