2024 IEEE Symposium on Security and Privacy (SP) | 979-8-3503-3130-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/SP54263.2024.00006

Multi-Instance Adversarial Attack on GNN-Based Malicious Domain Detection

Mahmoud Nazzal*, Issa Khalil§, Abdallah Khreishah*, NhatHai Phan*, and Yao Ma*
*New Jersey Institute of Technology, Newark, NJ 07102, USA
§Qatar Computing Research Institute (QCRI), Hamad Bin Khalifa University (HBKU), Doha, Qatar
Email: mn69@njit.edu, ikhalil@hbku.edu.qa,{abdallah,phan,yao.ma}@njit.edu

***Abstract*—Malicious domain detection (MDD) is an open security challenge that aims to detect if an Internet domain name is associated with cyber attacks. Many techniques have been applied to tackle this problem, among which graph neural networks (GNNs) are deemed one of the most effective approaches. GNN-based MDD employs domain name system (DNS) logs to represent Internet domains as nodes in a graph, dubbed domain maliciousness graph (DMG) and trains a GNN model to infer the maliciousness of Internet domains by leveraging the maliciousness of already identified ones. As this method heavily relies on the "publicly" accessible DNS logs to build DMGs, it creates a vulnerability for adversaries to manipulate the features and edges of their domain nodes within these graphs. The current body of literature primarily focuses on threat models that involve manipulating individual adversary (attacker) nodes. Nonetheless, adversaries usually create numerous domains to accomplish their attack objectives, aiming to reduce costs and evade detection. Hence, they aim to remain undetected across as many domains as possible. In this work, we call the attack that manipulates several nodes in the DMG concurrently *a multi-instance evasion attack*. To the best of our knowledge, this type of attack has not been explored in the prior art. We present both theoretical and empirical evidence to show that the existing single-instance evasion techniques for GNN-based MDDs are inadequate to launch multi-instance evasion attacks. Therefore, we propose an inference-time, multi-instance adversarial attack, dubbed MintA, against GNN-based MDD. MintA optimizes node perturbations to enhance the evasiveness of a node and its neighborhood. MintA only requires black-box access to the target model to launch the attack successfully. In other words, MintA does not require any knowledge of the MDD model's parameters, architecture, or information on non-adversary nodes. We formulate an optimization problem that satisfies the attack objectives of MintA and devise an approximate solution for it. We evaluate MintA on a state-of-the-art GNN-based MDD technique using real-world data, and our experiments demonstrate an attack success rate of over 80%. The findings of this study serve as a cautionary note for security experts, highlighting the vulnerability of GNN-based MDD to practical attacks that can impede the effectiveness and advantages of this approach.**

***Index Terms*—Adversarial attack, malicious domain detection, DNS logs, inference time attack.**

1. Introduction

Internet domains form a foundation for adversaries to launch various cyber attacks. For example, adversaries can use Internet domains to disseminate malware [1], facilitate command and control (C&C) communications [2], and host scam, phishing, and brand squatting web pages [3]. Malicious domain detection (MDD) refers to the problem of deciding whether a given Internet domain is used to launch malicious activities. Considerable efforts have been made in developing various MDD methodologies [4]. Among the different MDD methods, those that utilize domain name system (DNS) data possess unique advantages, such as scalability [5], rich domain features, and public accessibility [4]. Existing DNS-based MDD methods can be broadly categorized into classification-based and inference-based approaches [6]–[8]. Classification-based MDD relies on local domain features [9], [10]. In contrast, inference-based MDD integrates relations among domains and their local properties to achieve timely and accurate inference of domain maliciousness [5]–[7], [11]–[16].

In this paper, we focus on the inference-based MDD approach given its superior performance compared with other methods [7], [14]–[16]. Inference-based MDD is based on the *guilt-by-association* principle [13], [17], [18], i.e., if a domain is connected to a group of known malicious domains, then it is likely to be malicious as well. The inference-based MDD process comprises two primary phases. The initial phase involves creating a domain association graph known as the "domain maliciousness graph (DMG)." DNS data presents a valuable resource for entities engaged in MDD[1] to construct the DMG. The second phase involves utilizing an inference technique to identify the maliciousness of unknown domains in the DMG by using a small set of labeled domains as a reference. A DMG (e.g., Fig. 1) can be either a homogeneous graph with domain nodes and their relationships or a heterogeneous graph with multiple node and link types. For example, a DMG may consist of domain and client nodes [11], [19], [20], domain and IP address nodes [12], [13], or domain, IP address, and client nodes [7], [14]–[16].

Regarding the inference phase of MDD, techniques like belief propagation, graph neural networks (GNNs), or het-

[1] Let us refer to a party performing MDD as an *MDD entity*.

DOI 10.1109/SP54263.2024.00006

erogeneous graph neural networks (hetGNNs) are typically employed. Belief propagation is a commonly utilized inference method. However, its accuracy drops significantly when the labeled training data is insufficient, and it cannot make inferences regarding isolated domain nodes [21]. Recent research has explored more advanced inference methods using GNNs and hetGNNs [7], [14]–[16], [22], [23]. GNNs have been shown to offer significant advantages over belief propagation. Firstly, GNNs employ data-driven training to learn mechanisms for message passing and aggregation across graph nodes in an end-to-end fashion [24]. Secondly, GNNs are more adaptable to larger graphs with complex structures not initially seen [25]. Moreover, GNNs demonstrate superior performance when the labeled training data is limited [7], [14]–[16].

Despite the effectiveness of GNN-based models in accurately and quickly detecting malicious domains and extensive research on their vulnerability to adversarial attacks [26]–[28], multi-instance attacks have not been fully explored in the context of GNN-based MDD. In this scenario, an adversary (attacker) can only manipulate the features and relationships of the domains under its control (e.g., the orange subgraph in Fig. 1) to craft an adversarial attack to bypass the MDD model. This is possible due to the heavy reliance of MDD models on publicly available DNS logs to construct their DMGs, leaving room for malicious intervention.

Prior approaches and their limitations. The current body of literature primarily focuses on threat models that involve manipulating individual adversary nodes. However, adversaries usually create many domains to accomplish their attack objectives, aiming to reduce costs and evade detection. Hence, they aim to remain undetected across as many domains as possible. Existing approaches either target a specific node for evasion (e.g., [26], [27], [29]–[32]) or attack the model's overall performance through untargeted (availability) attacks (e.g., [33]–[36]). We demonstrate that repeatedly applying single-instance attacks to achieve multi-instance evasion is suboptimal and degenerates the attack's overall performance. On the other hand, availability attacks are easy to detect and do not achieve the evasion goal of the adversary.

Overview of our approach. Based on the previous discussion, we have formulated the adversarial attack in GNN-based MDD as a multi-instance evasion attack. To execute this attack, we design MintA as a multi-instance attack algorithm. MintA first constructs a surrogate model using only black-box access to the target model. It is worth noting that MintA is practical as it does not require knowledge of the target model's parameters, architecture, or other nodes in the DMG. After constructing the surrogate model, MintA employs it to identify the suitable feature and edge perturbations that will collectively evade the adversary nodes while minimizing the effect on other nodes in the DMG. This is achieved through a two-objective optimization problem at each adversary node. The first objective aims to maximize the model's loss at each adversary node to avoid detection, while the second objective aims to maximize the model's loss at the neighboring nodes to evade detection. To obtain an approximate solution, we maximize the weighted average of these two objectives. Once the appropriate edge/feature perturbations are identified, MintA implements them by making name and domain resolution edits to its nodes. This is achieved through domain registration services and IP resolution edits.

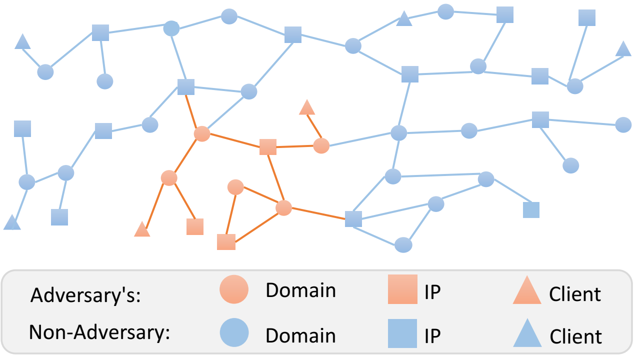


Figure 1. Based on DNS logs, the MDD entity constructs a DMG containing the adversary's subgraph (in orange) along with non-adversary subgraphs (in blue).

Summary of contributions. Our contribution in this study includes: (i) Demonstrating the vulnerability of GNN-based MDD and proposing MintA, a novel algorithm for the multi-instance adversarial attack that only requires black-box access to the target model and with no knowledge of the DMG graph beyond the adversary nodes. To the best of our knowledge, this is the first practical attack against GNN-based MDD. (ii) We present a method for implementing the optimized adversarial perturbations by simple domain name and IP-resolution manipulations. (iii) We conduct extensive experiments on real-world data to evaluate the effectiveness of our proposed attack. Our results show an attack success rate of over 80%. Also, MintA is shown to bypass outlier detection and graph purification-based defenses.

Paper outline. Section 2 presents relevant preliminaries. The threat model of adversarial attack in MDD is presented in Section 3. Section 4 presents the proposed MintA algorithm. Experiments and results are presented in Section 5. Section 6 summarizes related work. We present a discussion in Section 7 with the conclusions in Section 8.

2. Background

Graph Neural Networks. Graph neural networks (GNNs) [37] extend deep learning to graph data by using neural network layers for handling messages passing across graph edges and their aggregation. A GNN transforms graph information into a set of low-dimensional node embeddings calculated based on local node features and graph topology. GNNs have achieved state-of-the-art performances in a wide set of applications ranging from fraud detection to drug discovery [38]. A heterogeneous graph or a heterogeneous information network (HIN) is a graph with multiple node and/or edge types. Heterogeneous GNNs (hetGNNs) extend

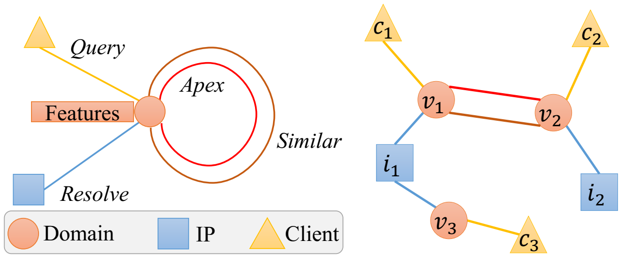
GNNs to HINs. Some hetGNNs project the HIN on a graph space to eliminate heterogeneity [39], while others decompose a given HIN into meta-paths[2] which are then encoded to get node representations [40], [41]. In a variety of applications, hetGNNs achieve state-of-the-art performances [42]–[45].

GNN-based MDD inference. The state-of-the-art MDD approaches employ heterogeneous DMGs [7], [14]–[16], [46]. Still, they differ in the types of nodes and edges assumed in their DMGs and the corresponding ways of achieving message passing and aggregation. First, HinDom [14] uses meta-paths with HINs of clients, domains, and IP addresses excluding node attributes. Subsequently, this setting is extended to attributed HINs. Node attributes are obtained as character-level domain properties in HGDom [22] and the 21 FANCI domain properties [47] in DeepDom [46]. Similarly, Zhang et al. [15] define a HIN of domains, IP addresses, and clients, where node type-aware feature transformation and edge type-aware message aggregation are employed. More recent works such as [16] and [8] adopt attention mechanisms to achieve message passing and aggregation. These approaches show the potential of GNNs, particularly hetGNNs, in achieving timely and accurate MDD.

From DNS logs to DMG. DNS is a key resource for constructing DMGs. As an example, Fig. 2(a) shows the network schema of a heterogeneous DMG. Domain, IP address, and client node types are included where domain features such as the 21 FANCI features [47] serve as the domain node attributes. Also, the following edge types exist.

- domain-*query*-client: linking a client node to a domain node it queries
- domain-*apex*-domain: linking two domain nodes if they share the same apex domain
- domain-*resolve*-IP: linking a domain node to an IP node it resolves to
- domain-*similar*-domain: linking two domain nodes if the n-gram character-level similarity between their names exceeds a preset threshold of 0.8 [48]

Let us consider an example of using DNS to construct a heterogeneous DMG according to the above network schema. Consider domain names *"www.b.rwth-aachen.de"*, *"writes.bnxd.rwth-aachen.de"*, and *"dekh1her76avy0qnelivijwd1.ddns.net"* represented by domain-type nodes v_1, v_2, and v_3, respectively, as shown in Fig. 2(b). Let us assume further that a given DNS log shows that the domains v_1, v_2, and v_3 resolve to IP addresses i_1, i_2, and i_1, respectively, and they are queried by clients c_1, c_2, and c_3, respectively. Recalling the four edge types mentioned above, the following edges exist. First, v_1 and v_3 are linked to i_1 whereas v_2 is linked to i_2, all with *resolve* edges. Second, v_1 through v_3 are linked to c_1 through c_3 with *query* edges, respectively. Third, v_1 is connected to v_2 with a *similar* edge due to their high character-level similarity and with an *apex* edge since they share the same apex address (*"rwth-aachen.de"*). Therefore, there is a direct mapping between the contents of DNS logs and the shape of the constructed DMG.



(a) (b)

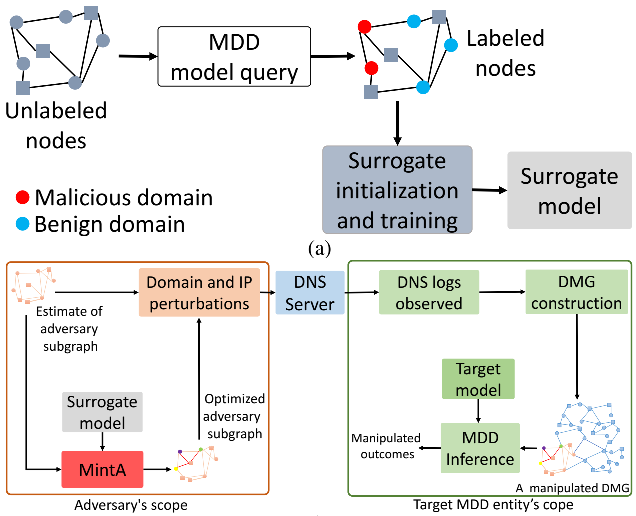
Figure 2. (a): Network schema of a heterogeneous DMG showing node and edge types. (b): An example DMG formed by associations extracted from DNS logs.

3. Threat Model

We characterize the threat model in terms of the adversary's goals, knowledge, and capabilities. The goal of the adversary is to evade the detection of its nodes by the MDD model with budgeted perturbations. As for knowledge, the adversary has partial knowledge of its actual subgraph residing in the DMG (the orange-colored subgraph in Fig. 1). Namely, the adversary knows its nodes (domains and IP addresses) but does not know the topology or the node features in its actual subgraph in the DMG. This information requires knowledge of the edge types, and node attributes the target MDD entity assumes. However, the adversary can still assume the usage of commonly used edge types and node attributes to obtain an estimate of its actual subgraph.

While different GNN-based MDD models may have different node and edge structures in their DMGs, they often utilize common edge types, such as domain-*similar*-domain or domain-*resolve*-IP, that are derived from DNS logs (e.g., [8], [14]–[16], [22], [46]). However, the surrogate model employed to optimize the perturbations is a simplified homogeneous model that does not differentiate between edge types. Additionally, common node features, like whether a domain name contains digits or has a "www" prefix, are utilized by multiple works in the field [8], [15], [16], [22], [46]. Thus, an adversary can assume the existence of these commonly used edges and features, even without precise knowledge of the DMG structure. In our problem formulation, we assume the adversary knows only an estimate of its subgraph. We hypothesize that perfectly knowing the remaining portions of the DMG and incorporating them into the solution may improve the attack's success, but the improvement would not be significant. This is because adversary nodes exhibit denser mutual connections among themselves compared to their connections with non-adversary nodes, as established in the following section. Moreover, the adversary does not know the target model's architecture, parameters, or training mechanism. It can only query the target model for a certain number of domain nodes to obtain labeled data to train a surrogate model. Thus, this is a black box attack. As for capabilities, the adversary can

[2] A meta-path is a composition of relations linking two nodes.



(a)

(b)

Figure 3. An overview of the attack; surrogate model training in (a), and crafting the attack in (b).

only manipulate specific editable node features and edges in its subgraph by modifying the names of its domains and their IP resolution relationships, as shown in the next section.

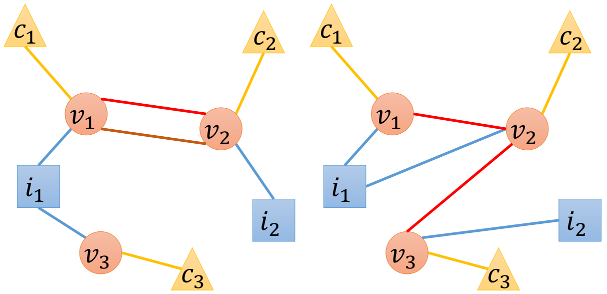
A general overview of the proposed attack setting is presented in Fig. 3. The adversary queries the target MDD model to obtain labeled data and trains a surrogate model, as shown in Fig. 3(a). Then, the adversary constructs an estimate of its subgraph in the DMG assuming the existence of commonly used edge types and node features. With this subgraph estimate, the adversary uses the surrogate model along with the proposed MintA (detailed in the next section) to obtain optimized feature (and/or edge) perturbations to the nodes in its subgraph, as shown in Fig. 3(b). Next, the adversary implements these perturbations by manipulating its domains' names and IP resolutions, as shown in Section 4.1. The modifications made by the adversary will be observable in the DNS logs, and eventually, the desired subgraph will emerge in the DMG constructed by the MDD entity.

Table 1. FLIPPING A FEATURE VECTOR BY A SIMPLE NAME EDIT.

Feature	*before*	*after*
Has a WWW prefix	1	0
Contains single-character subdomains	1	0
Is exclusive prefix repetition	0	1
Contains digits	0	1

4. The Proposed Attack

In this section, we first present how an adversary can implement intentional perturbations in a target DMG through DNS. Then, we analyze the conditions for an adversary to craft a successful adversarial attack. Next, we present MintA as an algorithm for optimizing the perturbations applied to the adversary's nodes.



(a)

(b)

Figure 4. An example DMG constructed based on DNS in (a) and the required perturbed version in (b).

4.1. Practical implementation of feature and edge perturbations

An Internet domain encompasses numerous features. However, altering some of these attributes can prove challenging, if not impossible. For example, changing the top-level domain (TLD) of a given domain from ".com" to ".edu" is not possible without institutional accreditation which is required for registering a domain name with this TLD. Among the 21 FANCI features, we select the following features as editable: *Domain name length*, *Has a www prefix*, *Contains a single-character subdomain*, *Is exclusive prefix repetition*, and *Contains digits*. As for edges, we select the domain-*resolve*-IP, domain-*apex*-domain, and domain-*similar*-domain edges [14]–[16], [22], [46] as editable.

To illustrate how an adversary can perturb edges and features in its subgraph on a DMG by domain name and IP resolution manipulations, let us recall the example domain names *"www.b.rwth-aachen.de"*, *"writes.bnxd.rwth-aachen.de"*, and *"dekh1her76avy0qnelivijwd1.ddns.net"* and their respective node representations, v_1, v_2, and v_3 appearing in Fig. 4(a). Next, let us consider feature perturbation on v_1 as an example with binary features for simplicity. For v_1, a 4-bit binary representation of the features *Has a www prefix*, *Contains a single-character subdomain*, *Is exclusive prefix repetition*, and *Contains digits* is [1100] since v_1 has a www prefix (1), contains a single-character subdomain (1), does not have an exclusive prefix repetition (0), and does not contain digits (0). The domain name owner can flip the feature vector to be [0011] by changing its name to *"bnxd3.bnxd3.rwth-aachen.cis"*[3]. Table 1 lists the feature vectors of this domain before and after the modification.

Assume that the adversary's goal is to modify the topology of the constructed graph shown in Fig. 4(a) into the one in Fig. 4(b). This can be implemented as follows.

- Swap the *resolve* edges between v_2 and v_3: by changing the resolution of domain v_2 to IP i_1, and that of v_3 to IP i_2.
- Drop the *similar* edge between v_1 and v_2: by changing node v_1's domain name from *"www.b.rwth-aachen.de"* to

[3] This modified name has no www prefix (0), does not contain a single-character subdomain (0), has an exclusive prefix repetition (1), and contains digits (1), thus the feature vector is [0011].

"www.b.sokj-bbchin.de" thereby dropping the character-level similarity between v_1 and v_2 to below 0.8.
- Add an *apex* edge between v_2 and v_3: by changing node v_3's domain name from *"dekh1her76avy0qnelivijwd1.ddns.net"* to *"dekh1her76avy0qnelivijwd1.ddns.rwth-aachen.de"* so that both share the same "rwth-aachen.de" apex address.

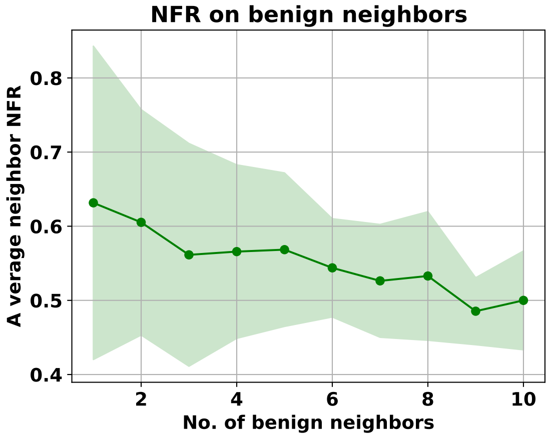


Figure 5. The impact of evading mutually-isolated adversary nodes on their benign neighbors (NFR: negative flip rate).

4.2. Multi-instance adversarial attack

In the following, we identify the requirements for an adversary to perform a successful adversarial attack against GNN-based MDD models.

1. The adversary owns multiple domains. To maintain agility in the face of individual domain blacklisting or takedown, adversaries usually operate across multiple domains [49]. This belief is supported by empirical evidence, such as the study conducted by Bahnsen et al. [50], which analyzed a practical phishing attack dataset and identified three adversaries owning 19, 106, and 309 domains, respectively, to host a total of 102, 1,007, and 7,927 phishing URLs. In addition, Hao et al. [51] advocate for registering multiple domains in bulk as a means for adversaries to reduce costs.

2. Interconnected adversary domains for efficient evasion in bulk. Interconnected domains allow the adversary to conduct attacks across those domains while maintaining a low profile. Conversely, isolated domains necessitate an individualized approach to attacking them, which can be more easily detectable. Such attacks may impact non-adversary domains connected to the adversary's domains, which exist as nodes in a DMG but are neither known nor exploited by the adversary. To investigate this scenario, we conduct the following experiment.

We randomly sample 100 domain nodes from a given test graph such that they are not mutually connected while they have edges with other nodes. These domain nodes are regarded as the adversary's nodes. Next, we use the integrated gradients adversarial attack (IG-ADV) proposed by Wu et al. [30] to evade the detection of each adversary domain individually. We then assess the effects of attacking each domain node on its benign neighbors on the graph. We quantify the impact of such an approach using the average neighbor negative flip rate (NFR). NFR represents the percentage of benign neighbors that turn malicious due to the evasion attempt of the malicious node. Fig. 5 shows the average neighbor NFR for domain nodes of the same number of benign neighbors. As depicted in the figure, an adversarial attack aimed at evading detection on a specific domain node has a detrimental impact on its benign neighbors. This negative effect makes the attack more easily detectable, compelling adversaries to avoid using mutually isolated domains. Instead, it is advisable for an adversary to have a subgraph of connected domains, as highlighted in orange in Fig. 1.

3. No Interference among adversary domains. It is essential to ensure that an attack aimed at evading detection on a specific domain node should not undermine the evasion strategy employed by other connected domain nodes. We make the observation that perturbations at a given adversary node optimized to maximize the loss function at this node to evade the detection may contradict the evasion of other adversarial nodes in its k-hop neighborhood. We carry out a detailed analysis to demonstrate this observation in Appendix A.1. This observation asserts that attempting to evade the detection of individual nodes belonging to the adversary, without considering the coordinated evasion of its entire subgraph, as typically done in existing targeted adversarial attacks, undermines the adversary's objective of evading detection altogether.

The following experiment is conducted to practically examine the above observation. We consider an adversary with 100 domain nodes. Then, we apply the IG-ADV attack [30] as a representative of targeted attacks on each domain node individually and gradually till attacking all the nodes. Meanwhile, we calculate the attack success rate (ASR) as the percentage of adversary domain nodes converted from malicious into benign[4]. Fig. 6 illustrates the relationship between the average ASR and the number of attacked nodes. While a single adversarial attack can successfully evade detection for a single node with an ASR of over 80%, it remains unclear whether these attacks can still be effective when targeting multiple connected nodes. As the number of attacked nodes increases, the overall ASR decreases significantly, indicating that individual attacks can harm the evasion attempts of their connected nodes.[5]

The preceding discussion asserts that the current adversarial attack methods are insufficient to fulfill the attackers' needs to carry out effective multi-node detection evasive

[4] This differs from our definition of the ASR in the other experiments, where it is the ratio of malicious domains evaded from the overall number of malicious domains.

[5] It is noted that IG-ADV is operated with the same constraints either when attacking single or multiple nodes. Thus, the degradation in ASR is due to overlooking the impact on connected nodes and is not due to constraints.

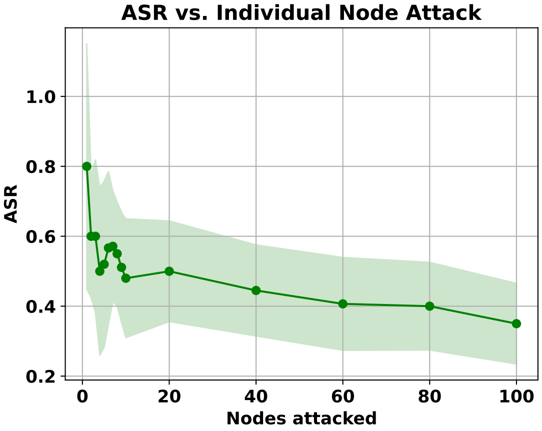


Figure 6. The attack success rate (ASR) of attacked nodes with sequentially applying a targeted adversarial attack.

attacks. Therefore, we devise a two-objective optimization problem for this multi-instance detection evasive attack. This problem aims to determine the best feature perturbation at a specific node i, denoted as $\Delta \boldsymbol{X'}_i$, which would help evade the detection of both node i and its neighboring nodes $j \in \mathcal{N}(i)$, where $\mathcal{N}(i)$ is the set of direct neighbors to i. In this study, we acknowledge the possibility of an adversary strategically optimizing perturbations across all nodes to achieve their attack objectives. Nevertheless, our approach focuses on optimizing perturbations at individual nodes, specifically node i, and assessing their impact on its direct neighbors, denoted as j. We find this strategy adequate for crafting a coordinated attack across the adversary's nodes because targeting the direct neighbors of node i allows the attack impact to propagate effectively throughout the graph.

Since the GNN-based MDD model is unknown to adversaries, we use a 2-hop linearized graph convolutional network (GCN) surrogate model [52]. This surrogate, known as the simplified GCN model, is commonly used in adversarial attacks [27], [53]–[55] due to its simplicity and the transferability of attacks conducted on it to different GNN architectures. This surrogate model has the same task as the target model (node classification) and is trained on a different dataset that is labeled by querying the target model. The sole similarity between the actual and surrogate models lies in the graph task they carry out, which is node classification. The surrogate model is shown to work well through validation, and attacking it is expected to transfer to the unknown target MDD model.

Let us use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote the DMG constructed by the MDD entity where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is their edges. Equivalently, $\mathcal{G}$ can be written as $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ where $\boldsymbol{A}$ is the adjacency matrix and $\boldsymbol{X}$ is the node attribute matrix. The adversary knows only its node set which we denote by $\mathcal{V}' \subset \mathcal{V}$ and some edges connecting them $\mathcal{E}' \subset \mathcal{E}$. Let us denote by $\mathcal{G}' = (\boldsymbol{A'}, \boldsymbol{X'})$ the adversary subgraph with adjacency matrix $\boldsymbol{A'}$ and node feature matrix $\boldsymbol{X'}$. We can write messages at the second layer of the surrogate model as $\boldsymbol{H'}^{(2)} = \hat{\boldsymbol{A}'}^2 \boldsymbol{X'} \boldsymbol{W}$, where $\hat{\boldsymbol{A}'} = \boldsymbol{D'}^{-\frac{1}{2}}(\boldsymbol{A'} + \boldsymbol{I})\boldsymbol{D'}^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix, $\boldsymbol{D'}$ is a diagonal matrix of node degrees [56], and $\boldsymbol{W}$ denotes the coefficients of the surrogate model. For simplicity, let $\boldsymbol{B}$ denote $\hat{\boldsymbol{A}'}^2$. Further, let us assume a sigmoid loss function, and ignore it in the loss calculation as done in [27]. Thus, the change in the loss function of the surrogate model at node i due to a perturbation $\Delta \boldsymbol{X}_i$ can be written as follows.

$$\Delta \mathcal{L}_i \left(\boldsymbol{A'}, \Delta \boldsymbol{X'}_i; \boldsymbol{W}, i\right) = \|\boldsymbol{B} \Delta \boldsymbol{X'}_i \boldsymbol{W}\|_2^2. \quad (1)$$

Similarly, the change in loss at node $j \in \mathcal{N}(i)$ is:

$$\Delta \mathcal{L}_j \left(\boldsymbol{A'}, \boldsymbol{X'} + \Delta \boldsymbol{X'}_i; \boldsymbol{W}, j\right) = \|\boldsymbol{B}(\Delta \boldsymbol{X'}_i + \boldsymbol{H'}_j)\boldsymbol{W}\|_2^2, \quad (2)$$

where $\mathcal{L}_i$ ($\mathcal{L}_j$) is the loss function value at node i (j), and $\boldsymbol{H'}_j$ is the message at node j.

Considering a given adversary's node i, and its direct neighbors $j \in \mathcal{N}(i)$ in its neighborhood set $\mathcal{N}(i)$ known to the adversary (in its subgraph), the formulation of the proposed feature optimization problem is expressed in (3). It is noted that i may have other neighbors known to the MDD entity but not known to the adversary.

$$\Delta \boldsymbol{X'}_i^* = \underset{\Delta \boldsymbol{X'}_i}{\operatorname{argmax}} \; \alpha \Delta \mathcal{L}_i \left(\boldsymbol{A'}, \boldsymbol{X'} + \Delta \boldsymbol{X'}_i; \boldsymbol{W}, i\right) + \sum_{j \in \mathcal{N}(i)} \beta \frac{1}{d_j} \Delta \mathcal{L}_j \left(\boldsymbol{A'}, \boldsymbol{X'} + \Delta \boldsymbol{X'}_i; \boldsymbol{W}, j\right), \; \forall \; i \in \mathcal{V}', \quad (3)$$

where α and β are weighted average parameters to control the trade-off between maximizing the local loss at node i and maximizing the loss at the neighbors $j \in \mathcal{N}(i)$.

A solution to the problem in (3) is obtained by maximizing a weighted average of $F_1 = \Delta \mathcal{L}_i \left(\boldsymbol{A'}, \boldsymbol{X'} + \Delta \boldsymbol{X'}_i; \boldsymbol{W}, i\right)$ and $F_2 = \sum_{j \in \mathcal{N}(i)} \frac{1}{d_j} \Delta \mathcal{L}_j \left(\boldsymbol{A'}, \boldsymbol{X'} + \Delta \boldsymbol{X'}_i; \boldsymbol{W}, j\right)$. With the use of the surrogate model, the magnitude of the change in loss $\Delta \mathcal{L}_i$ due to $\Delta \boldsymbol{X'}_i$ can be written as $\|\mathcal{L}_i(\boldsymbol{A'}, \boldsymbol{X'} + \Delta \boldsymbol{X'}_i; \boldsymbol{W}, i) - \mathcal{L}_i(\boldsymbol{A'}, \boldsymbol{X'}; \boldsymbol{W}, i)\| = \|\boldsymbol{B} \boldsymbol{X'}_i \boldsymbol{W}\|_2^2$. The optimization of F_1 with respect to $\Delta \boldsymbol{X'}_i$ can be written as follows.

$$\underset{\Delta \boldsymbol{X'}_i}{\operatorname{argmax}} \; F_1 = \underset{\Delta \boldsymbol{X'}_i}{\operatorname{argmax}} \; \|\boldsymbol{B} \Delta \boldsymbol{X'}_i \boldsymbol{W}\|_2^2 . \quad (4)$$

Similarly, we can view optimizing F_2 as follows.

$$\underset{\Delta \boldsymbol{X'}_i}{\operatorname{argmax}} \; F_2 = \underset{\Delta \boldsymbol{X'}_i}{\operatorname{argmax}} \sum_{j \in \mathcal{N}(i)} \|\boldsymbol{B} \left[\Delta \boldsymbol{X'}_i + \boldsymbol{H'}_j\right] \boldsymbol{W}\|_2^2 . \quad (5)$$

From (4), and Corollary 1.1 in Appendix A.1, an optimal $\Delta \boldsymbol{X'}_i$ for maximizing F_1 is the one that maximizes the sum of its inner products with the columns in the model coefficient matrix $\boldsymbol{W}$. Equivalently, it is a perturbation vector that maximizes $\boldsymbol{\Phi}_i = \boldsymbol{W}^T \Delta \boldsymbol{X'}_i$, where $\boldsymbol{\Phi}_i$ is equal to $\boldsymbol{W}^T$. Then, equivalently, an optimal $\Delta \boldsymbol{X'}_i$ is the one that maximizes the quantity $\|\boldsymbol{\Phi}_i \boldsymbol{X'}_i\|_2^2$. So the problem of perturbation optimization becomes a matrix-vector inner product maximization problem, which we can solve using Theorem 1, detailed below.

Algorithm 1 MintA-feature perturbation.

```
Input: An adversary subgraph $\mathcal{G}' : (\boldsymbol{A'} \in \mathbb{R}^{n\times n}, \boldsymbol{X'} \in \mathbb{R}^{n\times k})$, a feature perturbation budget $k_f$, and a set of editable node features $p$.
Output: A modified adversary subgraph $\mathcal{G}'^* : (\boldsymbol{A'}, \boldsymbol{X'}^*)$
1: Obtain labeled data by querying the target model.
2: Train a surrogate model.
3: Initialize total feature perturbation $\Delta\boldsymbol{X'} \leftarrow \mathbf{0}$, $i = 1$.
4: While $i \leq n$ AND $\|\Delta\boldsymbol{X'}\|_2 \leq k_f$
5:     Obtain an optimal perturbation $\Delta\boldsymbol{X'}^*_i$ according to (7)
6:     Modify the editable features in $\boldsymbol{X'}_i$ ($p$) to best match $\boldsymbol{X'}^*_i = \boldsymbol{X'}_i + \Delta\boldsymbol{X'}^*_i$
7:     Update $\Delta\boldsymbol{X'} = \Delta\boldsymbol{X'} + \Delta\boldsymbol{X'}^*_i$.
8:     Increment $i$.
9: return $\boldsymbol{X'}^* = \boldsymbol{X'} + \Delta\boldsymbol{X'}$
```

Theorem 1. *The following problem:*

$$\begin{aligned} &\underset{\Delta\boldsymbol{X'_i}}{\operatorname{argmax}} \ \|\boldsymbol{\Phi}_i \Delta\boldsymbol{X'_i}\|_2^2, \\ &s.t. \ \|\Delta\boldsymbol{X'_i}\|_2^2 = \epsilon, \end{aligned} \tag{6}$$

where ϵ is a threshold representing the budget of the perturbation, has a closed-form solution, which is the eigenvector corresponding to the largest eigenvalue of $\boldsymbol{\Phi}_i^T\boldsymbol{\Phi}_i$.

The proof of Theorem 1 is in Appendix A.2. According to Theorem 1, to solve (4), an optimal $\Delta\boldsymbol{X'}_i$ with respect to F_1 is the principal eigenvector of the matrix $\boldsymbol{\Phi}_i\boldsymbol{\Phi}_i^T = \boldsymbol{W}^T\boldsymbol{W}$, denoted by $\boldsymbol{e}_i$. Also, as shown in our analysis of the need for a coordinated subgraph attack in Appendix A.1, to solve (5) for a given $j \in \mathcal{N}(i)$, $\Delta\boldsymbol{X'}_i$ with respect to F_2, is the principal eigenvector of the matrix $\boldsymbol{\Phi}_j\boldsymbol{\Phi}_j^T$, where $\boldsymbol{\Phi}_j = (\boldsymbol{W} - \boldsymbol{H'}_j)$. Let us denote this solution by $\boldsymbol{e}_j$. Since $\boldsymbol{e}_i$ optimizes F_1 and $\boldsymbol{e}_j$ optimizes $F_2 \ \ \forall\, j \in \mathcal{N}(i)$, we can approximately meet both objective functions by a perturbation which is the weighted average of these solutions as shown in (7).

$$\Delta\boldsymbol{X'}^*_i = \alpha\boldsymbol{e}_i + \sum_j \beta d_j \boldsymbol{e}_j. \tag{7}$$

The perturbation obtained in (7) maximizes the weighted average of the loss functions at the node itself and its neighbors where α and β control their relative importance, respectively.

The adversary possesses nodes that are present in the DMG, and these nodes may be connected to non-adversary nodes. Such neighboring non-adversary nodes are susceptible to the perturbations made by their adversary neighbors. Therefore, it is worthwhile to examine the consequences of the adversary's perturbations on the non-adversary neighbors of their nodes. Proposition 1 compares this impact to situations where attacks are optimized for individual nodes.

Proposition 1. *Consider an adversary node i, connected to adversary nodes $j \in \mathcal{N}(i)$ and a non-adversary node l. On node l, the effect of a perturbation optimized by maximizing the loss over the node i and its direct neighbors $j \in \mathcal{N}(i)$ is smaller than the effect of optimizing the loss on only i.*

Algorithm 2 MintA-edge perturbation.

```
Input: An adversary subgraph $\mathcal{G}' : (\boldsymbol{A'} \in \mathbb{R}^{n\times n}, \boldsymbol{X'} \in \mathbb{R}^{n\times k})$, an edge perturbation budget $k_e$.
Output: A modified adversary subgraph $\mathcal{G}'^* : (\boldsymbol{A'}^*, \boldsymbol{X'})$
1: Obtain labeled data by querying the target model.
2: Train a surrogate model.
3: Initialize the number of edge flips $i = 0$.
4: While $i \leq k_e$
5:     Select the node pair having the maximum value of the average of $F_1$ and $F_2$ appearing in (4) and (5).
6:     Edge flip: edit the name or change the IP resolution to implement the edge edit.
7:     Increment $i$.
8: return $\boldsymbol{A'}^*$
```

The proof of Proposition 1 is in Appendix A.3.

4.3. The proposed MintA algorithm

The above perturbation optimization forms the foundation of our proposed MintA attack. The proposed MintA algorithm is described in Algorithm 1. First, the adversary trains a surrogate model (Steps 1-2). Then, it constructs an estimate of its attributed adversary subgraph. After that, the proposed attack is performed collaboratively on adversary nodes by optimizing their perturbations according to (7) and within a given feature perturbation budget k_f (Steps 3-5). Next, the optimal feature perturbation is approximated by manipulating the editable node features to best fit the desired optimized values (Step 6). The final modified feature matrix of the adversary nodes is obtained accordingly. MintA is also applicable to edge perturbations as well. This can be done by selecting edge edits that can best match the average of the objective functions in (7). Algorithm 2 presents the edge perturbation steps.

Table 2. A SUMMARY OF RESEARCH QUESTIONS AND ANSWERS.

Q	Property investigated	Key Result
1	Effectiveness (adversary nodes)	High ASR & Low NFR
2	Stealthiness (non-adversary nodes)	Low ASR & Low NFR
3	Robustness: outlier detection	High robustness
4	Robustness: graph purification	High robustness
5	Baseline comparisons	Higher ASR & Lower NFR
6	Costs	Low and scalable

5. Experiments

We conduct experiments to evaluate MintA's performance. The source code and data are available on the link: https://github.com/mahmoudkanazzal/MintA. Our evaluation aims to answer the questions summarized in Table 2.

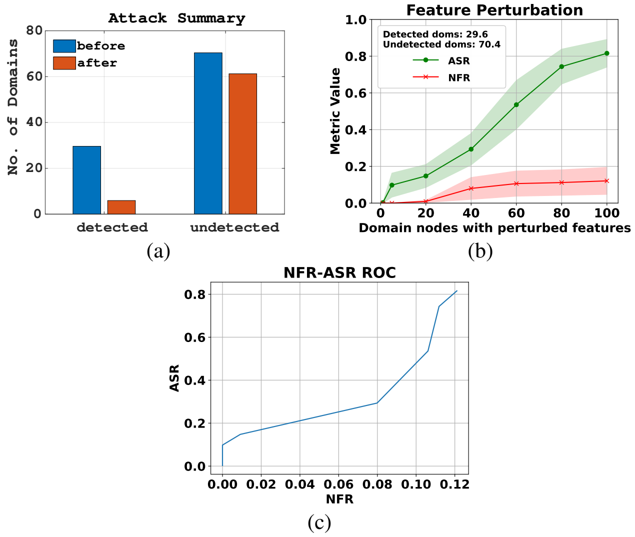
5.1. The setup and dataset

In our experiments, we consider Sun et al.'s algorithm [46] as a target GNN-based MDD model. The approach in

Table 3. KEY STATISTICS OF THE DATASET USED (CLIENT AND IP NODES HAVE NO LABELS).

Node Type	Eidsiva	PDNS	Total
Clients	12035	0	12035
Domains (total)	12035	465,905	477940
Domains (benign)	5,000	4,963	9,963
Domains (malicious)	5,000	20,354	25,354
IPs	8000	73,593	81593

[46] treats MDD as a semi-supervised node classification problem. The DMG in [46] is a heterogeneous graph created according to the network schema in Fig. 2(a) with the 21 FANCI features [47] as its domain node attributes. It is noted that [46] and all the other works in [7], [11], [12], [14]–[16] do not share datasets or artifacts as their graphs have private enterprise data. Therefore, we obtain a heterogeneous DMG with a similar structure by merging a domain-IP graph [48] with the Eidsiva enterprise graph, obtained from Eidsiva Bredband (a broadband telecom operator in Norway), collected by [57], and available at [58]. Table 3 lists key statistics of the merged dataset. It is noted that we use another dataset from [59] to query the target MDD model for obtaining data to train the surrogate model. By doing that, the surrogate model is trained independently from the training and testing DMGs of the target model. Experiments are conducted on a Lambda GPU Workstation with 128 GB of RAM, two GPUs each of 10 GB RAM, and an I9 CPU of 10 cores at a clock speed of 3.70 GHz.



(a)

(b)

(c)

Figure 7. With feature perturbation and *created adversary subgraphs*; attack summary when 100 domains are attacked in (a), ASR, NFR, and ROC when less than 100 domains are attacked in (b) and (c), respectively.

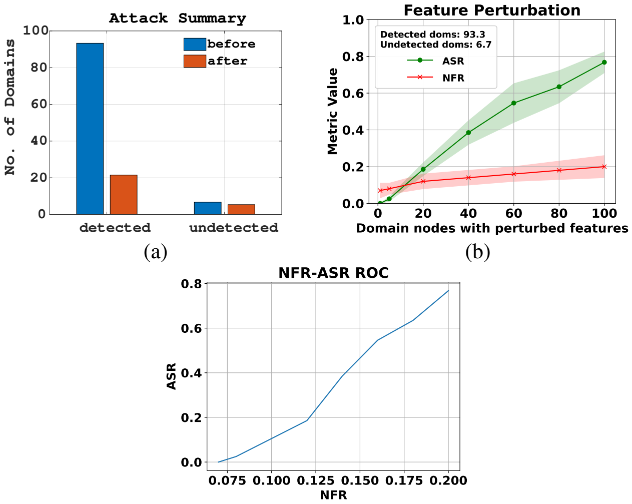
To conduct experiments with the proposed attack, it is essential to model the adversary's intervention in the DMG using what we refer to as the "adversary's subgraph". While subgraphs of domains owned by real adversaries would be ideal for this purpose, unfortunately, such data is not accessible. As a result, we resort to the following modeling approaches and incorporate them into our experiments.

- A *created adversary modeling approach:* We create a set of domain nodes, which will serve as the adversary's nodes (Registered in Feb. 2023 and intended to persist for a few months.) The MDD entity then utilizes the DNS logs to construct a DMG that incorporates these nodes. We use a free service provided by https://profreehost.com to create and host ULRs associated with the registered domain names, which will be dedicated to emulating the adversary's domain nodes. This process entails the creation of URLs, their free hosting, and the linkage of domain names to these URLs. By doing so, the adversary can construct its subgraph based on the knowledge of these domain names and their respective host IP addresses. It is noteworthy that this approach closely reflects the constraints faced by actual adversaries. However, it has some limitations as the target MDD might not accurately classify the URLs associated with these domains as malicious. To address this limitation, we also employ the following additional complementary adversary modeling approach.
- A *sampled adversary modeling approach:* In a given inference DMG, we acquire adversary nodes by randomly sampling connected nodes solely from the set of domain nodes that are labeled as malicious. We incorporate these adversary nodes into the test set to enable the MDD model to infer their malicious nature. This approach effectively models the adversary as a collection of interconnected malicious nodes.

5.2. Performance evaluation of MintA

In the following experiments, we address Q1: Does MintA meet the adversary's goals? We evaluate the performance of MintA in terms of two metrics. First is the attack success rate (ASR) which is the percentage of undetected malicious adversary domains due to the attack. Second, is a negative flip rate (NFR) which is the percentage of undetected adversary domains before applying MintA that are classified as malicious after we conduct MintA. This represents the side effect of the attack. It is noted that for the adversary to maximize the effectiveness of the attack, all of its domains need to be attacked (i.e., 100 in our case). Nonetheless, in our experiments, we examine the ASR and NFR for the cases of attacking less than 100 domains. We also present the ASR-NFR trade-off in a ROC curve. We repeat each experiment for 30 trials and report the average values of these metrics with their 95% confidence intervals. In each trial, we consider a different realization of the MDD model, the adversary nodes, and the attack.

5.2.1. Feature perturbation. In this experiment, we record the ASR and NFR after the adversary activates the feature perturbation attack (Algorithm 1). We set the feature perturbation budget k_f to the sum of dynamic ranges of editable features (domain name length feature range is 5 and binary features' range is 1, so the budget is 5.38 times the number of nodes involved in the attack). First, we consider the *created adversary subgraph*. We found that before activating



(a) (b)

(c)

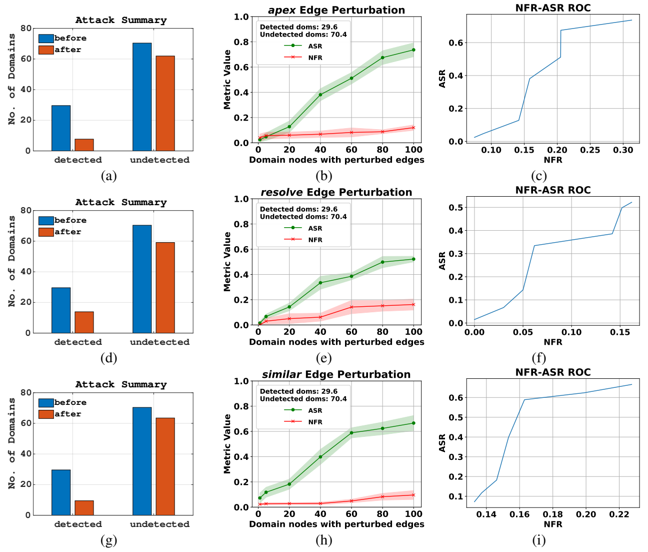
Figure 8. With feature perturbation and *sampled adversary subgraphs*; attack summary when 100 domains are attacked (a), ASR, NFR, and ROC when less than 100 domains are attacked in (b) and (c), respectively.

the attack, around 29.6% of the domains created on average are detected as malicious by the target MDD model, and the remaining 70.4% on average are not detected[6]. The attack summary is shown in Fig. 7(a). It shows that the attack evades the detection of 23.7 (0.80 × 29.6) adversary domains on average out of the 29.6% detectable adversary domains. The attack results in the detection of around 9.15 (0.13 × 70.4) domains on average from the undetectable domains. This shows that the attack is successful because the number of evaded domains is significantly larger than the ones detected after launching MintA. For the cases of attacking less than 100 domains, the ASR and NFR are shown in Fig 7(b) and the ROC curve is in Fig. 7(c).

Next, we repeat the above experiment with *sampled adversary subgraphs*. Before launching the MintA attack, we found out that 93.3% of the domains sampled, on average, are detected as malicious by the target MDD model, and the remaining 6.7% on average are not detected. For this experiment, the attack summary shown in Fig. 8(a) shows that the attack evades the detection of 71.8 (0.77 × 93.3) adversary domains on average out of the 93.3% detectable adversary domains. The attack results in the detection of about 1.3 (0.2 × 6.7) domains on average from the undetectable domains. Therefore, this attack is successful and is even more successful than in the case of the *created adversary subgraph* since more domains are evaded and fewer are made detectable. For the cases of attacking less than 100 domains, the ASR and NFR are shown in Fig. 8(b) and the ROC curve is in Fig. 8(c).

5.2.2. Edge perturbation. In this experiment, we evaluate edge perturbation attacks (Algorithm 2). We allocate the budget for edge perturbation, denoted as k_e, to be equivalent to the number of edges that will be swapped among the adversary domain nodes involved in the attack. This decision is based on the fact that the adversary possesses knowledge and control solely over the edges connected to its own nodes. We select the *apex*, *resolve*, and *similar* edges to perturb. Perturbing these edges can be implemented as specified in Section 4.1.

First, we consider the *created adversary subgraph*. The results with *apex* edge perturbation are shown in the top row of Fig. 9. For this attack, the attack summary plot is shown in Fig. 9(a). This plot shows that the attack evades the detection of 21.9 (0.74 × 29.6) adversary domains, on average, out of the 29.6% detectable adversary domains. The attack results in the detection of around 8.5 (0.12 × 70.4) domains, on average, from the undetectable domains. This shows that the attack is successful because the number of evaded domains is significantly larger than the ones detected from the undetectable domains. Compared with feature perturbation, the attack has less ASR and NFR. For the cases of attacking less than 100 domains, the ASR and NFR are shown in Fig 9(b) and the ROC curve is in Fig. 9(c).



(a) (b) (c)

(d) (e) (f)

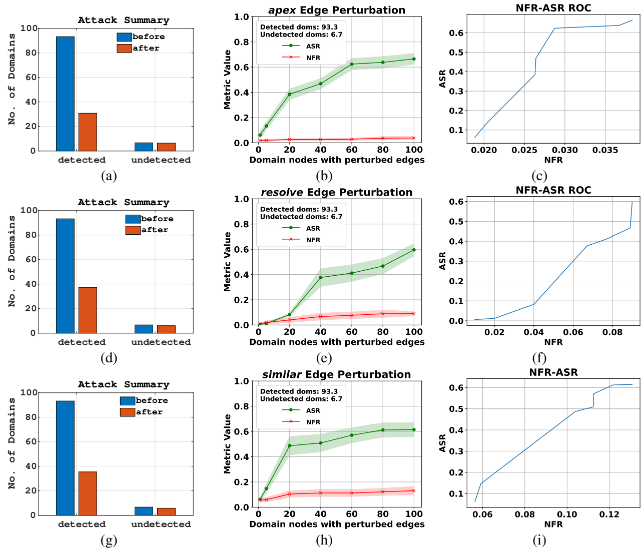
(g) (h) (i)

Figure 9. With *apex*, *resolve*, and *similar* edge perturbation and *created adversary subgraphs*; attack summary when 100 domains are attacked, ASR, NFR, and ROC when less than 100 domains are attacked, in rows 1, 2, and 3, respectively.

The *resolve* edge perturbation results are shown in the second row of Fig 9. Looking at its attack's summary graph in Fig 9(d), it evades the detection of 15.6 (0.53 × 29.6) adversary domains, on average, out of the 29.6% detectable adversary domains. The attack results in the detection of around 10.56 (0.15 × 70.4) domains, on average, from the undetectable domains. This attack is marginally successful. Nonetheless, it is less effective than feature and *apex*-edge attacks. The reason is that the adversary has limited IP addresses, and it can only use these addresses. This restriction limits the effectiveness of the attack. For the cases of

[6] This rationale is justifiable since these newly created domains do not currently host real cyber-attacks. As a result, they lack exploitable associations, such as connections with compromised clients, that could potentially expose their existence.

attacking less than 100 domains, the ASR and NFR are shown in Fig 9(e), and the ROC curve is in Fig. 9(f). Finally, the *similar* edge perturbation results are shown in the last row of Fig 9. Considering the attack summary in Fig 9(g), this edge perturbation performs better than the *resolve* edge (evades 20.1 domains on average (0.68 × 29.6), and causes the detection of 7.0 domains on average (0.1 × 70.4) and is less efficient than the *apex* edge case. For the cases of attacking less than 100 domains, the ASR and NFR are shown in Fig 9(h) and the ROC curve is in Fig. 9(i).

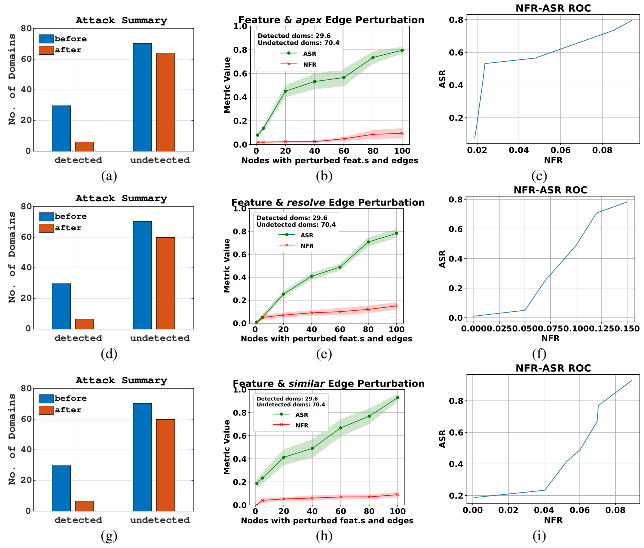


(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 10. With *apex*, *resolve*, and *similar* edge perturbation and *sampled adversary subgraphs*; attack summary when 100 domains are attacked, ASR, NFR, and ROC when less than 100 domains are attacked, in rows 1, 2, and 3, respectively.

Next, we repeat the previous experiment with *sampled adversary subgraphs*. The results with *apex* edge perturbation are in the top row of Fig. 10. For this case, the attack summary is in Fig. 10(a). It shows that, on average, the attack evades the detection of 62.5 (0.67 × 93.3) adversary domains out of the 93.3% detectable adversary domains. The attack results in the detection of around 0.26 (0.04 × 6.7) domains, on average, from the undetectable domains. Thus, the attack is successful in terms of ASR and NFR. Compared to feature perturbation, this attack has a similar ASR but a much lower NFR. For the cases of attacking less than 100 domains, the ASR and NFR are shown in Fig 10(b) and the ROC curve is in Fig. 10(c).

The *resolve* edge perturbation results are shown in the second row of Fig 10. Looking at its attack summary graph in Fig 10(d), this attack evades the detection of 55.9 (0.6 × 93.3) adversary domains, on average. The attack results in the detection of around 0.6 (0.09 × 6.7) domains, on average, from the undetectable domains. Similar to the case with created domains, the *resolve* edge attack is relatively poorer than the *apex* edge attack. Nonetheless, it is not that poor compared to the case of created domains. This is because the adversary has access to more IP addresses since its domains are obtained by *sampled adversary subgraphs* which may *resolve* to more diverse IP addresses. Finally, the *similar* edge perturbation results are shown in the last row of Fig 10. Considering the attack summary in Fig 10(g), this attack evades 57.8 (0.62 × 29.6) domains, on average, while causing the detection of 0.87 (0.13 × 6.7) domains, on average. The performance here is slightly better than that of the *resolve* edge perturbation and is less competitive than that of the *apex* edge perturbation. For the cases of attacking less than 100 domains, the ASR and NFR are shown in Fig. 10(h) and the ROC curve is in Fig. 10(i).



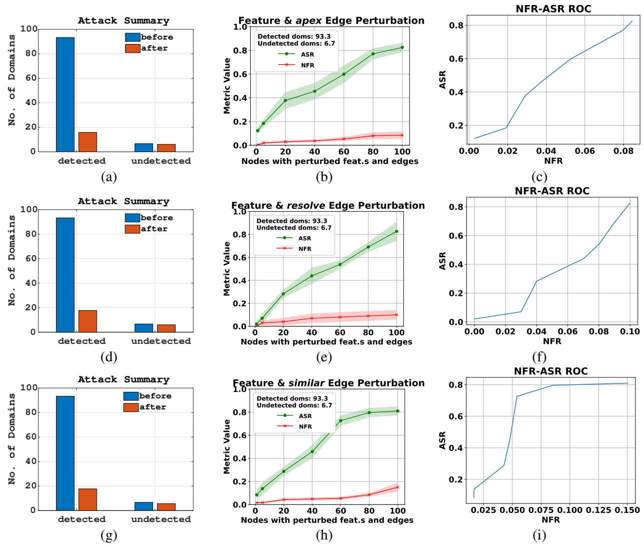
(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 11. With feature-*apex*, feature-*resolve*, and feature-*similar* edge perturbation and *created adversary subgraphs*; attack summary when 100 domains are attacked, ASR, NFR, and ROC when less than 100 domains are attacked, in rows 1, 2, and 3, respectively.

5.2.3. Joint perturbation.

In this experiment, we combine feature and edge perturbations (Algorithms 1 and 2). First, with created domains, the results with feature-*apex*, feature-*resolve*, and feature-*similar* perturbations are shown in the three rows of Fig. 11, respectively. According to the attack summary in Fig. 11(a), the feature-*apex* attack evades the detection of 23.67 (0.8 × 29.6) domains, on average, and causes the detection of 6.3 (0.09 × 70.4) domains, and is thus successful. Let us next consider the attack summary plot of the feature-*resolve* edge in Fig. 11. This attack evades 23.1 (0.78 × 29.6) domains, on average, and results in the detection of 10.6 (0.15 × 70.4) domains, on average. Thus, it is less successful compared to the previous attack. However, this attack is stronger than the case of feature and *resolve* edge attacks alone. Finally, the attack summary of feature-*similar* perturbation is in Fig. 11(g). This attack evades the detection of 27.2 (0.92 × 29.6) domains, on average, and only causes the detection of 6.3 (0.09 × 70.4) undetectable domains, on average. This attack is the strongest among the combined attacks and is more successful than individually attacking the features or *similar* edges.

Finally, we repeat the previous experiment with *sampled adversary subgraphs*. The results for feature-*apex*, feature-



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

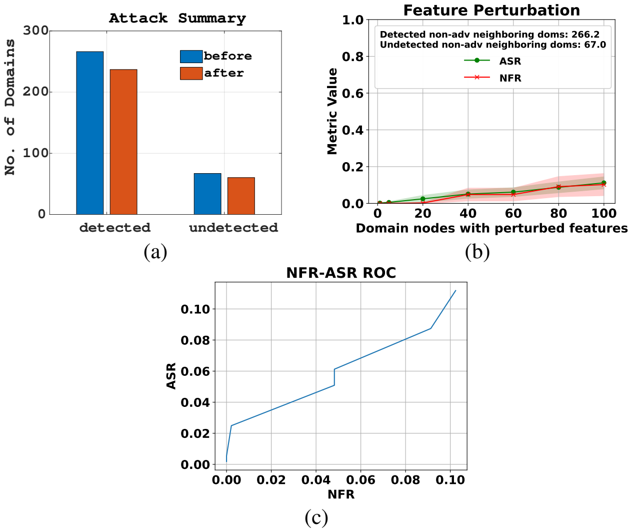
Figure 12. With feature-*apex*, feature-*resolve*, and feature-*similar* edge perturbation and *sampled adversary subgraphs*; attack summary when 100 domains are attacked, ASR, NFR, and ROC when less than 100 domains are attacked, in rows 1, 2, and 3, respectively.

resolve, and feature-*similar* perturbation attacks are shown in the three rows of Fig. 12, respectively. The feature-*apex* edge causes the evasion of 76.5 (0.82 × 93.3) domains on average while causing the detection of 0.53 (0.08 × 6.7) domains on average. The results of the other two scenarios are similar. In general, the performance with these perturbations is better than individually perturbing the features or edges. These results show the advantage of combining edge and feature perturbations.

The above experiments show that MintA is successful as a framework for adversarial attacks in MDD. This is validated by its ability to evade the detection of high numbers of adversary domains while leading to the detection of fewer undetectable adversary domains.

5.3. The impact of MintA on non-adversary nodes

An adversary's subgraph resides in a larger DMG formed by the MDD entity. To address Q2: "What is MintA's impact on non-adversary nodes?", we present the following experiment. We consider feature perturbations for the adversary's nodes. For each number of adversary nodes with perturbed features, we monitor the impact on non-adversary nodes residing in the same DMG being direct neighbors to adversary nodes. Fig. 13 presents the results of this experiment. As seen in the figure, both the ASR and NFR impacts of adversary node perturbation on non-adversary nodes increase when attacking more adversary nodes. Still, the attack summary plot shows that the attack results in evading the detection of 29 domains that were detectable before the attack (11%×266.2=29.38), while it causes around 7 benign domains (10%×66.7=6.67) to be detected as malicious. The results demonstrate that MintA's impact on non-adversary neighboring nodes is relatively moderate and inconspicuous. In other words, the effect is too subtle to raise suspicions of adversarial activity or compromise its stealthiness.



(a) (b)

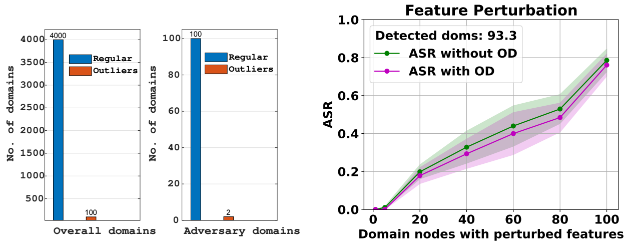
(c)

Figure 13. The effect of feature perturbations on non-adversary (direct) neighboring domains; attack summary in (a), and ASR, NFR, and ROC in (b) and (c), respectively.

5.4. The performance with outlier detection

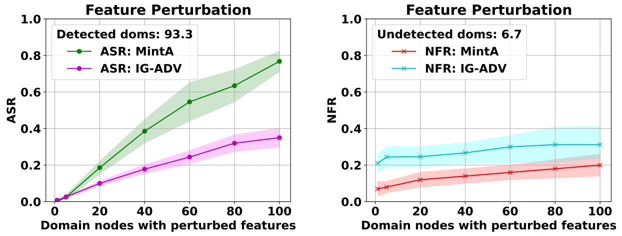
Recent literature considers the use of simple outlier detection as a basic defense mechanism to detect adversarial examples [60]. It is reasonable to assume that an MDD entity may apply outlier detection as a preliminary defense measure where domains identified as outliers are considered malicious, even before using the MDD model. Hence, it is important to quantify whether MintA causes the attacked nodes to appear as outliers. For this purpose, we address Q3: Can the domains perturbed by MintA be detected as outliers? through the following two experiments. In the first experiment, we sample a total of 4k nodes and randomly sample 100 of these to be the adversary nodes. Next, we perturb the features of all adversary nodes using Algorithm 1. Then, we consider the features of nodes in the whole set as a sample set and apply outlier detection to this set. We use isolation forest [61] as an outlier detection method. We adjust the parameters of outlier detection such that the number of outliers is around 100 nodes. To this end, we quantify how many of the detected outliers are data points (features of nodes) of the adversary nodes. In Fig. 14(a), we plot the detection histograms of the overall sample (left) and the adversary nodes (right). It is seen that only 2 of the 100 adversary nodes are detected from the overall 100 outliers in the sample set of data points. This is close to the overall percentage of outliers in the whole test set which is 2.5% (=100/4k). Thus, an outlier detection method does not distinguish the nodes perturbed by MintA from other nodes.

In the second experiment, We compare the performance of MintA against plain MDD to the case where an MDD



(a) (b)

Figure 14. (a) Histogram of outliers detected when setting the number of outliers to be 100 from a total set of 4k samples. (b) The ASR of MintA with and without outlier detection.



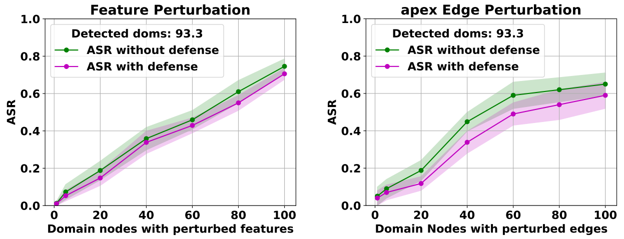
(a) (b)

Figure 16. ASR and NFR comparison of MintA with sequentially applying the IG-ADV attack [30].

entity employs outlier detection and identifies outlier nodes as malicious automatically before the use of the MDD model. Therefore, for this latter scenario, we subtract the ratio of outliers detected in the adversary nodes from the calculated ASR. We also adjust the outlier detection algorithm to produce about 100 outliers. Results are shown in Fig. 14(b). In this figure, it is evident that the MintA attack is only marginally affected by outlier detection.

5.5. Performance with graph purification defense

To examine the performance of MintA when the MDD entity applies graph purification defense, we raise Q4: Can the domains perturbed by MintA bypass graph purification-based defense? by considering the case of applying the GCN-Jaccard method [30]. This defense method is based on dropping Jaccard-dissimilar edges in a given graph suspecting their maliciousness. Hence, any domain node in the DMG dropped by this method is considered malicious. Fig. 15 shows the impact of this graph purification on feature and *apex* edge perturbations in parts (a) and (b), respectively. We only show *apex* perturbation since the *resolve* and *similar* edge perturbations perform similarly. The attack generally bypasses this defense in both scenarios. However, there is more degradation in the edge perturbation case compared to feature perturbation. This is reasonable since graph purification naturally affects the topology of the graph. Thus, it is more harmful to edge perturbations than to feature perturbations. These results validate that MintA is resilient to graph purification-based defenses.



(a) (b)

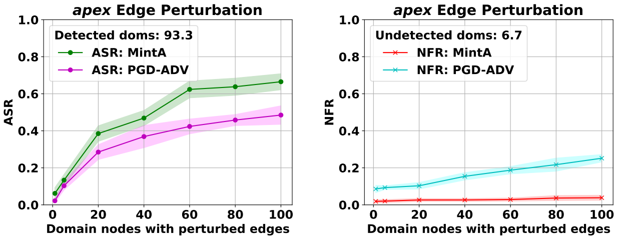
Figure 15. The impact of graph purification defense by the MDD entity on MintA's ASR with feature perturbation in (a) and *apex* edge perturbation in (b).

5.6. Comparison with targeted adversarial attacks

Here, we address Q5: What is the performance of MintA compared to a targeted adversarial attack? MintA is the only adversarial attack capable of handling multiple connected nodes, which is precisely its intended purpose. Therefore, we consider the closest baseline for comparison with MintA to be the repeated application of single-instance attacks. We have already presented an experiment on the ineffectiveness of this approach when considering the IG-ADV attack by Wu et al. [30] in Section 4.2. For completeness, we compare MintA to two examples of single-instance attack methods. It is noted that the ASR used in the experiment in Section 4.2 is the ratio of domains evaded from the attacked domains, whereas, in this section, we use the same ASR definition used in the other experiments, i.e., the ASR is the ratio of evaded domains to the total number of adversary's nodes (100 in our case), for consistency.

First, we present an experiment on feature perturbation. Fig. 16 compares the ASR and NFR of MintA and IG-ADV attack [30] where IG-ADV is applied individually on each node. IG-ADV node attack starts to have increases in ASR with increasing the number of nodes attacked. However, as more nodes are attacked, the increase in ASR diminishes. This is due to the aggregate effect of attacking multiple nodes without coordination as discussed in Section 4.2. As for NFR, the NFR with IG-ADV starts with a relatively high value compared to MintA. Next, both exhibit increases in NFR as more nodes are attacked. Still, the NFR values of MintA are less than those of IG-ADV.

In another experiment, we compare MintA's edge perturbation to that of the projected gradient descent topology attack (PGD-ADV) by Xu et al. [33], as a baseline attack method. Fig. 17 shows the results. This figure shows that the MintA attack is successful on the adversary nodes, whereas PGD-ADV is less effective. Also, the NFR of PGD-ADV is higher than that of MintA and the difference increases with an increased number of attacked nodes. These results show the advantage of MintA over targeted attacks focusing on a specific node, as MintA alleviates the negative effect of node perturbations on each other's evasiveness, as exhibited with these attacks.



(a) (b)

Figure 17. ASR and NFR comparison of MintA and sequentially applying PGD-ADV attack [33].

5.7. Empirical cost analysis

MintA entails primarily the following cost-effective expenses: **Preparation cost**: This involves querying the target model to label training data for the surrogate model, and in our case, we utilized 600 queries. Querying is typically facilitated through free API or website services. Surrogate training can be conducted using a standard or cloud computing platform. **Perturbation optimization cost**: This cost depends on the number of adversary nodes and can be handled by any computer or cloud server. **Perturbations implementation cost**: Implementing perturbations requires human effort in modifying domain name and/or IP resolution, which involves contacting domain registrants and hosting service providers. There may be potential fees depending on the subscription plans. To this end, to address Q6, which pertains to the time and memory costs of MintA, we measure the execution time (in seconds) and memory usage (in megabytes) for the processes of surrogate training and MintA optimization (Algorithm 1). We quantify memory usage using the *psutil* Python package and conduct 30 trials, varying the number of adversary nodes. The empirical costs are summarized in Table 4. As indicated, the time and memory costs for MintA's preparation and optimization are relatively low and can be afforded by any regular or cloud computing platform. The optimization time cost scales linearly with the number of adversary nodes, while the other costs remain relatively stable. These results demonstrate the scalability of MintA.

Table 4. Average execution time (in second) for preparation (t_p) and optimization (t_o), and the corresponding memory usages (in Mb) (m_p), and (m_o), respectively.

No. of Adv. nodes	t_p	t_o	m_p	m_o
50	2.3	4.2	62.7	58.5
100	2.3	22.1	61.0	64.7
200	2.2	28.4	61.1	64.7
500	2.2	94.7	61.0	64.9
1000	2.2	284.8	61.1	64.7

5.8. Additional experiments

In addition, we explore the effects of the following factors on the attack: the percentage of adversary nodes in a DMG, the complexity of the surrogate model utilized, and white-box model access. Detailed information and results of these experiments can be found in Appendix B.

6. Related Work

Adversarial attacks on GNNs. The vulnerability of GNN models to adversarial attacks is well-studied [62]. This includes their operation in node classification [34], graph classification [63], and link prediction [64] in various applications. However, the vulnerability of GNNs in MDD is not studied. Only attacks on node classification can be relevant to MDD as a node-label inference operation. Based on the attack scope, attacks are either targeted, i.e., focusing on a given target node, or untargeted (availability attacks) aiming to degrade the performance of the target model. Examples of untargeted attacks include using the gradient [33] and reinforcement learning [34] to add/remove edges. More recently, Ma et al. [35], [36] consider the problem of selecting the best nodes for crafting an attack in a given graph. These works address the node selection process, not optimizing the perturbation itself. On the other hand, examples of targeted attacks include reinforcement learning-based perturbation [26], absolute gradient FGA [29], integrated gradients [30], and greedy perturbation selection for maximizing the loss of a surrogate model as in Nettack [27]. More recently, the concept of anchor nodes has arisen. Those are important nodes in a graph such that flipping their connectivity to a given target node is sufficient to flip the model's outcome for this node. Anchor nodes are either found by searching for them in a given graph [28], or by creating and adding them to the graph [65], [66]. The novelty of MintA lies in handling simultaneous evasion of multiple connected nodes in a coordinated manner. Besides, using MintA to attack other GNN operations that use node features and graph edges can be a future extension.

Other works consider indirect adversarial attacks, where an adversary manipulates the neighbors of a given target node to *remotely* influence it [27], [67], [68]. Nettack introduces the concept of influencer attacks [27], while [67] employs a generative approach to manipulate multiple nodes connected to the target node. Also, [68] proposes an influencer attack that operates at a 2-hop distance from the target node, extending Nettack's approach. However, these works do not consider simultaneous attacks on a set of multiple connected nodes and overlook the significance of multi-instance attacks [68].

7. Discussion

Impact of the study. This study establishes a significant security threat against one of the most prominent MDD approaches, which has garnered attention from both academia [7], [14]–[16], [46] and industry [3], [17], [23], [69]. Moreover, it is worth noting that commercialized MDD tools like DomainTools with Maltego [70] explicitly mention adopting a graph-based approach but do not disclose their exact graph inference technique for proprietary reasons. In our experiments, we specifically focus on the approach presented in [46] as a representative case. However, it is essential to emphasize that MintA is applicable to any other GNN-based MDD method utilizing DNS logs. We demonstrate that the costs associated with the preparation, optimization, and implementation of MintA are reasonable and manageable under mild conditions.

Limitations of the study. The primary limitations of this study arise from several factors. Firstly, the scarcity of MDD datasets is a significant constraint due to concerns regarding potential security vulnerabilities found in enterprise data available through DNS logs. Additionally, the usage of patented GNN-based approaches within the industry restricted our ability to explore a broader range of comparison options. Another limitation stems from the absence of applicable adaptive graph purification defenses for hetGNNs, which could have enhanced the robustness of the MDD approach. Furthermore, the absence of actual adversary subgraphs necessitated the adoption of sampled and created adversary modeling approaches. Although these approaches were designed to mimic adversary behaviors, utilizing actual domains associated with a real adversary would have provided a more accurate representation of adversary subgraphs during the experiments.

Possible countermeasures and future work. Future research can explore different avenues to enhance the attack and defense methods in the context of state-of-the-art MDD approaches that leverage heterogeneous graphs and hetGNNs. One potential direction is to harness the heterogeneity of the DMGs to further improve the effectiveness of the attack. Conversely, another important extension is to develop efficient defense mechanisms to counter MintA's vulnerability that has been demonstrated. A promising direction for defense is to integrate and complement the knowledge gained from DNS logs to bolster the robustness of MDD, given that malicious actors can access these logs. It is worth noting that, despite MintA's stealthiness against outlier detection, exploring approaches that leverage additional information from DNS logs could enhance the defense against such attacks.

8. Conclusions

In this research, we investigate the vulnerability of current Graph Neural Network (GNN)-based Malicious Domain Detection (MDD) approaches to evasive adversarial attacks during inference. We demonstrate how adversaries can practically exploit the reliance of GNN-based MDD on DNS logs to carefully modify their domain nodes, effectively altering the resulting graphs constructed by MDD entities. To optimize these modifications for effective evasion, we propose a novel multi-instance adversarial attack called MintA. This attack is particularly suited for practical adversaries who own multiple interconnected domains, considering cost and stealthiness factors. Unlike existing adversarial attacks, MintA aims to simultaneously evade detection for as many adversary domain nodes as possible.

Our experiments reveal that MintA achieves remarkable success rates, exceeding 80%, when targeting a state-of-the-art GNN-based MDD algorithm with real-world data. Also, we demonstrate that MintA can bypass defense methods based on outlier detection and graph purification, underscoring its effectiveness in evading detection mechanisms.

Acknowledgment

This work is partially supported by grants NSF IIS-2041096 and NSF CNS-1935928.

References

[1] S. Abraham and I. Chengalur-Smith, "An overview of social engineering malware: Trends, tactics, and implications," *Technology in Society*, vol. 32, no. 3, pp. 183–196, 2010.

[2] H. R. Zeidanloo and A. A. Manaf, "Botnet command and control mechanisms," in *2009 Second International Conference on Computer and Electrical Engineering*, vol. 1. IEEE, 2009, pp. 564–568.

[3] M. Nabeel, I. M. Khalil, and T. Yu, "Brand squatting domain detection systems and methods," Jun. 23 2022, uS Patent App. 17/558,986.

[4] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through DNS data analysis," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[5] I. Khalil, B. Guan, M. Nabeel, and T. Yu, "Killing two birds with one stone: Malicious domain detection with high accuracy and coverage," *arXiv preprint arXiv:1711.00300*, 2017.

[6] M. Nabeel, I. M. Khalil, B. Guan, and T. Yu, "Following passive DNS traces to detect stealthy malicious domains via graph inference," *ACM Transactions on Privacy and Security (TOPS)*, vol. 23, no. 4, pp. 1–36, 2020.

[7] Y. Li, X. Luo, L. Wang, and Z. Xu, "DyDom: Detecting malicious domains with spatial-temporal analysis on dynamic graphs," in *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2021, pp. 283–290.

[8] Q. Wang, C. Dong, S. Jian, D. Du, Z. Lu, Y. Qi, D. Han, X. Ma, F. Wang, and Y. Liu, "HANDOM: Heterogeneous attention network model for malicious domain detection," *Computers & Security*, p. 103059, 2022.

[9] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[10] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, pp. 1–28, 2014.

[11] P. Manadhata, S. Yadav, P. Rao, and W. Horne, "Detecting malicious domains via graph inference," in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, 2014, pp. 59–60.

[12] I. Khalil, T. Yu, and B. Guan, "Discovering malicious domains through passive DNS data graph analysis," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 663–674.

[13] I. M. Khalil, B. Guan, M. Nabeel, and T. Yu, "A domain is only as good as its buddies: Detecting stealthy malicious domains via graph inference," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, 2018, pp. 330–341.

[14] X. Sun, M. Tong, J. Yang, L. Xinran, and L. Heng, "HinDom: A robust malicious domain detection system based on heterogeneous information network with transductive classification," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, 2019, pp. 399–412.

[15] S. Zhang, Z. Zhou, D. Li, Y. Zhong, Q. Liu, W. Yang, and S. Li, "Attributed heterogeneous graph neural network for malicious domain detection," in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2021, pp. 397–403.

[16] Z. Li, F. Yuan, Y. Liu, C. Cao, F. Fang, and J. Tan, "Heterogeneous graph attention network for malicious domain detection," in *International Conference on Artificial Neural Networks*. Springer, 2022, pp. 506–518.

[17] I. Khalil, T. Yu, and M. C. Dacier, "Method to identify malicious web domain names thanks to their dynamics," Jun. 9 2020, uS Patent 10,681,070.

[18] P. Xia, M. Nabeel, I. Khalil, H. Wang, and T. Yu, "Identifying and characterizing covid-19 themed malicious domain campaigns," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 209–220.

[19] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Segugio: Efficient behavior-based tracking of malware-control domains in large ISP networks," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2015, pp. 403–414.

[20] J. Lee and H. Lee, "GMAD: Graph-based malware activity detection by DNS traffic analysis," *Computer Communications*, vol. 49, pp. 33–47, 2014.

[21] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 685–693.

[22] X. Sun, J. Yang, Z. Wang, and H. Liu, "HGDom: heterogeneous graph convolutional networks for malicious domain detection," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[23] M. Nabeel, I. M. Khalil, T. Yu, and E. Choo, "Method and system for domain maliciousness assessment via real-time graph inference," Dec. 21 2021, uS Patent 11,206,275.

[24] Z. Zhang, F. Wu, and W. S. Lee, "Factor graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8577–8587, 2020.

[25] K. Yoon, R. Liao, Y. Xiong, L. Zhang, E. Fetaya, R. Urtasun, R. Zemel, and X. Pitkow, "Inference in probabilistic graphical models by graph neural networks," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 868–875.

[26] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *International conference on machine learning*. PMLR, 2018, pp. 1115–1124.

[27] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2847–2856.

[28] X. Zang, Y. Xie, J. Chen, and B. Yuan, "Graph universal adversarial attacks: A few bad actors ruin graph learning models," *arXiv preprint arXiv:2002.04784*, 2020.

[29] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," *arXiv preprint arXiv:1809.02797*, 2018.

[30] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples on graph data: Deep insights into attack and defense," *arXiv preprint arXiv:1903.01610*, 2019.

[31] H. Chang, Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, W. Zhu, and J. Huang, "A restricted black-box adversarial framework towards attacking graph embedding models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3389–3396.

[32] B. Wang, T. Zhou, M. Lin, P. Zhou, A. Li, M. Pang, C. Fu, H. Li, and Y. Chen, "Evasion attacks to graph neural networks via influence function," *arXiv preprint arXiv:2009.00203*, 2020.

[33] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," *arXiv preprint arXiv:1906.04214*, 2019.

[34] Y. Ma, S. Wang, T. Derr, L. Wu, and J. Tang, "Attacking graph convolutional networks via rewiring," *arXiv preprint arXiv:1906.03750*, 2019.

[35] J. Ma, S. Ding, and Q. Mei, "Towards more practical adversarial attacks on graph neural networks," *Advances in neural information processing systems*, vol. 33, pp. 4756–4766, 2020.

[36] J. Ma, J. Deng, and Q. Mei, "Adversarial attack on graph neural networks as an influence maximization problem," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 675–685.

[37] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[38] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[39] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, 2020, pp. 2704–2710.

[40] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022–2032.

[41] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proceedings of The Web Conference 2020*, 2020, pp. 2331–2341.

[42] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 2077–2085.

[43] S. Li, J. Yang, G. Liang, T. Li, and K. Zhao, "SybilFlyover: Heterogeneous graph-based fake account detection model on social networks," *Knowledge-Based Systems*, vol. 258, p. 110038, 2022.

[44] N. Jiang, F. Duan, H. Chen, W. Huang, and X. Liu, "MAFI: GNN-based multiple aggregators and feature interactions network for fraud detection over heterogeneous graph," *IEEE Transactions on Big Data*, 2021.

[45] J. Hu, T. Li, Y. Zhuang, S. Huang, and S. Dong, "GFD: A weighted heterogeneous graph embedding based approach for fraud detection in mobile advertising." *Security & Communication Networks*, 2020.

[46] X. Sun, Z. Wang, J. Yang, and X. Liu, "Deepdom: Malicious domain detection with scalable and heterogeneous graph convolutional networks," *Computers & Security*, vol. 99, p. 102057, 2020.

[47] S. Schüppen, D. Teubert, P. Herrmann, and U. Meyer, "FANCI: Feature-based automated NXDomain classification and intelligence," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1165–1181.

[48] U. Kumarasinghe, F. Deniz, and M. Nabeel, "PDNS-Net: A large heterogeneous graph benchmark dataset of network resolutions for graph learning," *arXiv preprint arXiv:2203.07969*, 2022.

[49] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamcraft: An inside look at spam campaign orchestration." in *LEET*, 2009.

[50] A. C. Bahnsen, I. Torroledo, L. D. Camacho, and S. Villegas, "Deepphish: simulating malicious AI," in *2018 APWG symposium on electronic crime research (eCrime)*, 2018, pp. 1–8.

[51] S. Hao, A. Kantchelian, B. Miller, V. Paxson, and N. Feamster, "PREDATOR: Proactive recognition and elimination of domain abuse at time-of-registration," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 1568–1579.

[52] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[53] J. Li, T. Xie, L. Chen, F. Xie, X. He, and Z. Zheng, "Adversarial attack on large scale graph," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 82–95, 2021.

[54] Y. Chen, H. Yang, Y. Zhang, K. Ma, T. Liu, B. Han, and J. Cheng, "Understanding and improving graph injection attack by promoting unnoticeability," *arXiv preprint arXiv:2202.08057*, 2022.

[55] Z. Zhu, C. Wu, M. Zhou, H. Liao, D. Lian, and E. Chen, "Resisting graph adversarial attack via cooperative homophilous augmentation," *arXiv preprint arXiv:2211.08068*, 2022.

[56] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[57] E. Rismyhr, "Graph representation of DNS-related data for detecting malicious actions," Master's thesis, NTNU, 2020.

[58] ——, "Eirikrismyhr/mis4900: Master's thesis - information security." [Online]. Available: https://github.com/eirikrismyhr/MIS4900

[59] C. Marques, S. Malta, and J. P. Magalhães, "DNS dataset for malicious domains detection," *Data in Brief*, vol. 38, p. 107342, 2021.

[60] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, "Detection of adversarial training examples in poisoning attacks through anomaly detection," *arXiv preprint arXiv:1802.03041*, 2018.

[61] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012.

[62] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International Journal of Automation and Computing*, vol. 17, pp. 151–178, 2020.

[63] H. Zhang, B. Wu, X. Yang, C. Zhou, S. Wang, X. Yuan, and S. Pan, "Projective ranking: A transferable evasion attack method on graph neural networks," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3617–3621.

[64] P. Bhardwaj, J. Kelleher, L. Costabello, and D. O'Sullivan, "Poisoning knowledge graph embeddings via relation inference patterns," *arXiv preprint arXiv:2111.06345*, 2021.

[65] J. Dai, W. Zhu, and X. Luo, "A targeted universal attack on graph convolutional network by using fake nodes," *Neural Processing Letters*, vol. 54, no. 4, pp. 3321–3337, 2022.

[66] X. Zang, J. Chen, and B. Yuan, "GUAP: Graph universal attack through adversarial patching," *arXiv preprint arXiv:2301.01731*, 2023.

[67] J. Chen, D. Zhang, Z. Ming, K. Huang, W. Jiang, and C. Cui, "GraphAttacker: A general multi-task graph attack framework," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 2, pp. 577–595, 2021.

[68] T. Takahashi, "Indirect adversarial attacks via poisoning neighbors for graph convolutional networks," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 1395–1400.

[69] S. B. Tirumala, F. Wu, and C. R. Johnson, "Scoring domains and IPS using domain resolution data to identify malicious domains and IPS," Dec. 20 2022, uS Patent 11,533,293.

[70] "Domaintools - maltego integration," https://www.domaintools.com/integrations/maltego/, accessed on July 15, 2023.

[71] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

Appendix A. Analysis and Proofs

A.1. Analysis of the need for a coordinated subgraph attack

Let us recall the formulations of the two objective functions F_1 and F_2 in (4) and (5), respectively. We can optimize $\Delta \boldsymbol{X'}_i$ to maximize F_1 and F_2 using Corollary 1.1.

Corollary 1.1. *The quantity* $\mathrm{F}_1 = \|\boldsymbol{B}\Delta\boldsymbol{X'}_i\boldsymbol{W}\|_2^2$ *is maximized by maximizing the sum of inner products* $\langle \Delta\boldsymbol{X'}_i, \boldsymbol{W}_k\rangle,\ \forall\ k \in \{1,\ldots,K\}$ *where* $\boldsymbol{W} \in \mathbb{R}^{n\times K}$.

Proof. Le us make use of the fact that $\Delta\boldsymbol{X'}_i$ only changes the feature matrix at the i-th row (corresponding to node i). Let us now quantify the effect of this change with the help of matrix form representation of the formulations of F_1 and F_2, as follows.

$$\mathrm{F}_1 = \|\boldsymbol{B}\Delta\boldsymbol{X'}_i\boldsymbol{W}\|_2^2 =$$
$$\left\| \begin{bmatrix} B_{11} & B_{12} & \ldots & B_{1n} \\ B_{21} & B_{22} & \ldots & B_{2n} \\ \vdots & \ddots & \ldots & \vdots \\ B_{n1} & \ldots & \ldots & B_{nn} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \Delta\boldsymbol{X'}_i \\ 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_1 \ldots \boldsymbol{W}_n \end{bmatrix} \right\|_2^2 , \tag{8}$$

where $B_{i,j}$ is the element in $\boldsymbol{B}$ at the i-th row and j-th column, and $\boldsymbol{W}_i$ is the i-th column in the model parameter matrix $\boldsymbol{W}$. With a few algebraic simplification steps, we can write.

$$F_1 = \left\| \begin{bmatrix} B_{1i}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_1\rangle \ldots B_{1i}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_n\rangle \\ B_{2i}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_1\rangle \ldots B_{2i}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_n\rangle \\ \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ B_{ni}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_1\rangle \ldots B_{ni}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_n\rangle \end{bmatrix} \right\|_2^2 . \tag{9}$$

It is clear now that maximizing F_1 requires maximizing the sum of inner products $\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_k\rangle,\ \forall\ k.$ □

Now, the effect of $\Delta\boldsymbol{X'}_i$ on the message at node $j \in \mathcal{N}(i)$ is

$$\Delta\boldsymbol{X'}_j = [B_{ji}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_1\rangle \ \ldots \ \ B_{ji}\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_n\rangle] . \tag{10}$$

This will be added to the aggregate message at the node. $j\,(\boldsymbol{H'}_j)$. So, the loss at node j will be changed as follows.

$$\Delta\mathcal{L}_j\left(\boldsymbol{A'},\boldsymbol{X'}+\Delta\boldsymbol{X'}_i;\boldsymbol{W},j\right) = \|\boldsymbol{B}\left[\Delta\boldsymbol{X'}_i+\boldsymbol{H'}_j\right]\boldsymbol{W}\|_2^2 . \tag{11}$$

Following Corollary 1.1 and (11) for maximizing $\mathcal{L}_j\ \forall\ j \in \mathcal{N}(i)$, or equivalently, maximizing F_2, this requires maximizing the inner products $\langle\Delta\boldsymbol{X'}_i,\boldsymbol{W}_k - \boldsymbol{H'}_j\rangle,\ \forall\ k \in \{1,\ldots,K\},\ j \in \mathcal{N}(i)$. It is clear that optimizing $\Delta\boldsymbol{X'}_i$ only considering F_1 will not meet maximizing F_2. This proves that adversarial perturbations applied at node i are not optimized to serve the evasion of node j.

A.2. The proof of Theorem 1

Proof. Introduce a Lagrange multiplier λ to the formulation in (6), for simplicity, let us use $\boldsymbol{X'}$ to denote $\Delta \boldsymbol{X'}_i$.

$$F = \|\boldsymbol{\Phi X'}\|^2 - \lambda \sum_j \boldsymbol{X'}_j^2 - \epsilon = 0.$$

Differentiating both sides:

$$\frac{\partial F}{\partial \boldsymbol{X'}_k} = \frac{\partial}{\partial \boldsymbol{X'}_{\mathrm{k}}} \sum_j \left(\sum_i \boldsymbol{\Phi}_{ij} \boldsymbol{X'}_i \right)^2 - \lambda \frac{\partial}{\partial \boldsymbol{X'}_{\mathrm{k}}} \sum_j \boldsymbol{X'}_j^2 = 0,$$

$$\frac{\partial}{\partial \boldsymbol{X'}_{\mathrm{k}}} \sum_j \left(\sum_i \boldsymbol{\Phi}_{ij} \boldsymbol{X'}_i \right)^2 = \lambda \frac{\partial}{\partial \boldsymbol{X'}_k} \sum_j \boldsymbol{X'}_j^2,$$

$$\sum_j \boldsymbol{\Phi}_{kj} \sum_i \boldsymbol{\Phi}_{ij} \boldsymbol{X'}_i = \lambda \boldsymbol{X'}_k.$$

A solution is the eigenvector of $\boldsymbol{\Phi}^T \boldsymbol{\Phi}$, since

$$\left\| \boldsymbol{\Phi X'}^* \right\|^2 = \lambda^2 \left\| \boldsymbol{X'}^{*2} \right\|.$$

Then, we select the eigenvector having the largest eigenvalue of $\boldsymbol{\Phi}^T \boldsymbol{\Phi}$ since this will maximize the objective function. □

A.3. The proof of Proposition 1

Proof. consider an adversary node i, connected to j adversary nodes and a non-adversary node l. The adversary knows i and j and is not aware of l. Let us compare the contribution of the perturbation applied at node i to the message collected at node l in cases where the optimization is done by maximizing the loss on node i, and the proposed case where the perturbation is done by maximizing a weighted average of the loss at node i and its direct adversary neighbors $j \in \mathcal{N}(i)$, in a coordinated fashion.

For the case of perturbation optimization by maximizing the loss at node i, from (4), and Corollary 1.1 in Appendix A.1, an optimal $\Delta \boldsymbol{X'}_i$ must maximize its inner products with the columns $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_n$ in the model parameter matrix $\boldsymbol{W}$. Next, we showed in Theorem 1 that this solution is the principal eigenvector of the matrix $\boldsymbol{W}^T \boldsymbol{W}$. This solution is denoted by $\boldsymbol{e}$ in equation (7). Let us use the notation $\boldsymbol{e}(\boldsymbol{W})$ to denote the solution. The magnitude of the effect of the perturbation at i received at node l can be written as follows.

$$\|m_{single}\| = \frac{1}{d_l} \|\boldsymbol{e}(\boldsymbol{W})\|, \tag{12}$$

where d_l is the degree of node l.

In the proposed MintA attack case, the magnitude of message contribution of the perturbation is.

$$\|m_{proposed}\| = \frac{1}{d_l} \|\alpha \operatorname*{argmax}_{\Delta \boldsymbol{X'}_i} F_1\| + \frac{1}{d_l} \|\beta \operatorname*{argmax}_{\Delta \boldsymbol{X'}_i} F_2\|. \tag{13}$$

This can be written as:

$$\|m_{proposed}\| = \|(\alpha \boldsymbol{e}(\boldsymbol{W}) + \beta \sum_i \frac{1}{d_j} \boldsymbol{e}(\boldsymbol{W} - \boldsymbol{H'}_j)\|. \tag{14}$$

Applying the triangle inequality,

$$\|m_{proposed}\| \leq \alpha \|(\boldsymbol{e}(\boldsymbol{W})\| + \beta \| \sum_i \frac{1}{d_j} \boldsymbol{e}(\boldsymbol{W} - \boldsymbol{H'}_j)\|. \tag{15}$$

Now, let us evaluate the ratio of $m_{proposed}$ to m_{single}.

$$\begin{aligned} \frac{\|m_{proposed}\|}{\|m_{single}\|} &\leq \frac{\alpha \|\boldsymbol{e}(\boldsymbol{W})\| + \beta \| \sum_j \frac{1}{d_j} \boldsymbol{e}(\boldsymbol{W} - \boldsymbol{H'}_j)\|}{\|\boldsymbol{e}(\boldsymbol{W})\|} \\ &= \alpha + \beta \frac{\| \sum_j \frac{1}{d_j} \boldsymbol{e}(\boldsymbol{W} - \boldsymbol{H'}_j)\|}{\|\boldsymbol{e}(\boldsymbol{W})\|}. \end{aligned} \tag{16}$$

The numerator in the quotient in (16) is the magnitude of the summation of multiple vectors of arbitrary directions. For these vectors to add up in magnitude, their directions need to be aligned. This means that the eigenvectors of the matrices $(\boldsymbol{W} - \boldsymbol{H'}_j))^T (\boldsymbol{W} - \boldsymbol{H'}_j))$ need to align $\forall\ j$. This requires these matrices to be scalar multiples of each other. This condition can be met only if the messages $(\boldsymbol{H'}_j)$ are equal. This requirement can not be met since different nodes can have different messages. Now, let us assume that the perturbation $\Delta \boldsymbol{X'}_i$ has the same perturbation magnitude in both cases (single node attack and the proposed MintA), and the solutions $\boldsymbol{e}$ are unit-norm eigenvectors to be scaled by the perturbation budget, and let us assume that nodes i and j have the same degree. Then, the maximum value of the quotient in (16) is $\alpha + \beta$ which is 1 since α and β are weighted average coefficients. Thus, we can write

$$\|m_{proposed}\| / \|m_{single}\| < \alpha + \beta = 1. \tag{17}$$

Therefore, $\|m_{proposed}\|$ is strictly less than $\|m_{single}\|$. □

Appendix B. Supplementary experiments and results

Impact of the adversary subgraph's share from the DMG. The experiments in Section 5 consider an adversary owning 100 domains and a DMG of 4000 nodes. To further evaluate MintA, we experiment with varying numbers of adversary nodes (from 50 to 1000). For each case, the ASR and NFR are recorded when all adversary nodes are attacked, averaged over 30 trials, and presented in Fig. 18.

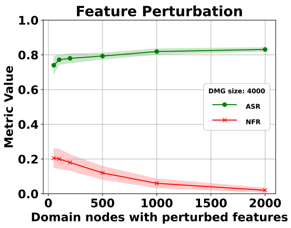
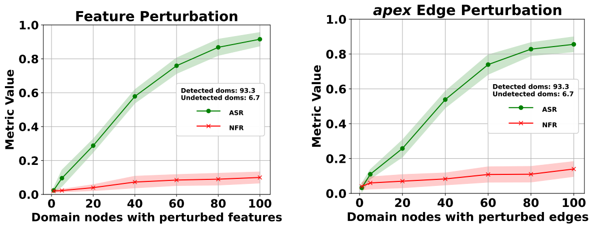


Figure 18. Average ASR and NFR versus the adversary's share on a DMG when all adversary nodes are involved in the attack.

From Fig. 18, in general, MintA is more successful with increasing its node share in the DMG, in terms of both ASR and NFR. However, the ASR increase and NFR decrease are not commensurate with the increase in the adversary's share



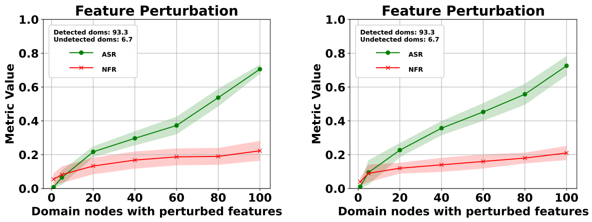
(a) (b)

Figure 19. Feature and *apex* edge performance with white-box model access in (a) and (b), receptively.

in the DMG. We attribute this behavior to the impact of messages fed to the adversary's nodes from non-adversary nodes (unknown to the adversary). This impact dramatically decreases as more nodes in the DMG belong to the adversary. This means that the effect of ignoring non-adversary messages decreases. This will continue till the limiting case where the adversary possesses the entire DMG. However, it is more practically sound to assume an adversary with a small portion of the DMG as it is not possible to intervene with the DMG construction at the MDD entity.

White-box model access. In the following experiment, we evaluate MintA in a white-box setting. We examine MintA with feature and *apex* edge perturbation attacks assuming access to the target MDD model parameters. Fig. 19(a) shows the performance with feature perturbation while Fig. 19(b) shows the performance with apex-edge perturbation. According to this figure, it is seen that white-box access to the model improves both attack metrics (ASR and NFR). However, the attack is still not perfect since the validity of the linearized model assumed in the formulations (and perturbation optimizations) is violated.

Different surrogates. To quantify the impact of the complexity and architecture of a surrogate model on MintA's performance, we repeat the feature perturbation attack (with a *sampled adversary subgraph*) with two surrogates; a homogeneous graph attention network (GAT) [71] and a heterogeneous GAT (hetGAT) that uses edge-aware message passing with GAT and assumes the same edge types used by the target model. Fig. 20 shows that both surrogates are inferior to the case of using a linearized GCN. This is because the validity of the linearized surrogate model becomes less accurate with using more sophisticated surrogates. However, the performance with the hetGAT surrogate is slightly better than that of the simple GAT one.



(a) (b)

Figure 20. Average ASR and NFR with a GAT surrogate and a hetGAT surrogate in (a) and (b), respectively.

Appendix C. Meta-Review

C.1. Summary

This paper proposes an adversarial evasion attack on graph-neural-network-based malicious domain detectors (MDD). The authors optimize node/edge perturbations on a surrogate MDD based on the subgraph of adversary nodes, and evaluate the effects of perturbations on the target MDD.

C.2. Scientific Contributions

- Identifies an Impactful Vulnerability
- Provides a Valuable Step Forward in an Established Field
- Independent Confirmation of Important Results with Limited Prior Research

C.3. Reasons for Acceptance

1) The paper demonstrates an impactful vulnerability in GNN-based MDDs. The authors propose a multi-instance adversarial attack that requires black-box access to the target MDD, and consider both outlier detection and graph purification as defense mechanisms.
2) The paper provides a valuable step forward in an established field, demonstrating that the effort to evade the detection at a certain adversary node can contradict the evasion of other adversary nodes. The authors formulate the evasion as a two-objective optimization problem that jointly maximizes the model's loss at each adversary node and its neighbors.
3) The paper designs and performs experiments under both synthetic and real-world settings to evaluate the efficacy of the proposed attack, demonstrating its feasibility.

C.4. Noteworthy Concerns

1) The methodology for the use of sampled and created adversary models lacks clarity, which is exacerbated by imprecise language and complicated evaluation setup. For example, the target malicious domain detection (MDD) model demonstrates inadequate performance in "created adversary" settings, where the detection rate for malicious domains is 30%, suggesting that the adversary has already achieved significant success with over 70% of their domains being misclassified as benign, without even conducting any form of attack. Furthermore, the "created adversary" experiments appear to conflate the ground truth with own labeling. Reviewers believe that a discussion on threats to the validity of the experimental results could be beneficial, which can clarify the methodological choices of the evaluation.

2) Threat model diverges from prior work (i.e., [30]) without motivation. Specifically, the assumptions to preserve the graph structure and feature statistics, which ensure that perturbations are unnoticeable, are violated. It is thus unclear whether IG adversarial attack can serve as an appropriate baseline given the different assumptions.
3) The paper does not discuss the relationship between the quality of the surrogate and attack success rate, e.g., how many queries the adversary should use to train their MDD.

Appendix D. Response to the Meta-Review

Below is a summary of our responses to the remaining concerns in the meta-review.

- Concern 1: During our experiments, we aim to model the adversary's intervention in the DMGs using what we refer to as the "adversary's subgraph". However, since actual adversary domain nodes are unavailable, we resort to two different modeling approaches. In the first approach which we call the *sampled adversary modeling*, we exclusively sample connected nodes from the ground-truth malicious domains in the test set. This allows us to simulate the behavior of an adversary without actual adversary domain data. In the second approach, which we call the *created adversary modeling*, we manually register domains to simulate the creation of adversary domains. We carefully adhere to practical constraints while creating these domains to provide a realistic representation of an adversary's behavior. Further details on these modeling approaches, including the motivations behind their usage and their limitations, can be found in the last two paragraphs of Subsection 5.1 (*The setup and dataset*). Additionally, in Section 7 (*Discussion*), we address the limitations of our study, including the absence of datasets containing domains of actual adversaries. These limitations are explained in detail in the second paragraph of that section.
- Concern 2: MintA maintains its stealthiness by manipulating only a small fraction, specifically 5 out of 45 elements, in the feature vector while preserving the overall statistics of the node features. In the experiment showcased in Subsection 5.4 (*The performance with outlier detection*), MintA demonstrates its ability to evade outlier detection, as it does not manipulate statistical features, which are crucial for isolation forest-based detection methods. In comparison to existing single-instance attack methods like IG-ADV [30] and PGD-ADV [33], which excel at attacking individual nodes but struggle against multiple connected nodes, MintA is specifically designed for multi-instance attacks. It outperforms these baselines when applied to connected nodes, as illustrated in Fig. 16 and Fig. 17. Further details explaining the ineffectiveness of IG-ADV with multiple-connected nodes are provided in Footnote 5. The rationale behind selecting IG-ADV and PGD-ADV as the closest baselines for comparison is discussed in the first paragraph of Subsection 5.6 (*Comparison with targeted adversarial attacks*).
- Concern 3: To facilitate ease of use and enable the analysis of input perturbations, we employ a simplified linearized 2-hop GCN as a surrogate model. During the surrogate model training, we query the target model for 600 domain nodes. It is essential to note that the training data for the surrogate is entirely distinct from the evaluation datasets to avoid any data leakage. The success of the attack using this surrogate model is an indicator of potential transferability to the target model. Although we evaluated more complex surrogate models, they were found to be inferior to the linearized GCN in terms of performance. To provide clarity on the surrogate model's quality, we include a clarification in the second paragraph of Section 3 (*Threat Model*). Additionally, experiments on the two additional surrogates are presented in Fig. 20 and the last paragraph of Appendix B.