GNN4IFA
Interest Flooding Attack Detection With Graph Neural Networks

Agiollo, Andrea; Bardhi, Enkeleda; Conti, Mauro; Lazzeretti, Riccardo; Losiouk, Eleonora; Omicini, Andrea

DOI
10.1109/EuroSP57164.2023.00043

Publication date
2023

Document Version
Final published version

Published in
Proceedings - 8th IEEE European Symposium on Security and Privacy, Euro S and P 2023

2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-6512-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/EuroSP57164.2023.00043

GNN4IFA: Interest Flooding Attack Detection With Graph Neural Networks

Andrea Agiollo*¶, Enkeleda Bardhi†¶, Mauro Conti‡§, Riccardo Lazzeretti†,
Eleonora Losiouk‡, Andrea Omicini*
**University of Bologna,* {andrea.agiollo, andrea.omicini}@unibo.it
†*Sapienza University of Rome,* {bardhi, lazzeretti}@diag.uniroma1.it
‡*University of Padua,* {conti, elosiouk}@math.unipd.it
§*Delft University of Technology,* M.Conti@tudelft.nl
¶*Co-first authors*

Abstract—**In the context of Information-Centric Networking, Interest Flooding Attacks (IFAs) represent a new and dangerous sort of distributed denial of service. Since existing proposals targeting IFAs mainly focus on local information, in this paper we propose GNN4IFA as the first mechanism exploiting complex non-local knowledge for IFA detection by leveraging Graph Neural Networks (GNNs) handling the overall network topology.**

In order to test GNN4IFA, we collect SPOTIFAI, a novel dataset filling the current lack of available IFA datasets by covering a variety of IFA setups, including ∼40 heterogeneous scenarios over three network topologies. We show that GNN4IFA performs well on all tested topologies and setups, reaching over 99% detection rate along with a negligible false positive rate and small computational costs. Overall, GNN4IFA overcomes state-of-the-art detection mechanisms both in terms of raw detection and flexibility, and – unlike all previous solutions in the literature – also enables the transfer of its detection on network topologies different from the one used in its design phase.

Index Terms—**Network Security, Interest Flooding Attacks, Graph Neural Networks, Emerging Networks**

1. Introduction

The concept of Information-Centric Networking (ICN) has its roots in the TRIAD project [1] (2000), whose objective was to replace the IP layer with a content-centric layer. Afterward, the research community proposed various ICN architectures: among them, Named-Data Networking (NDN) [2], [3] is considered the most promising. NDN switches the communication paradigm from host-based – i.e., where the data is – to content-based – i.e., what the data is. With this switch, NDN removes IP addressing and refers to the data using application-level names. In particular, NDN users look for content by name via interest requests. Data is delivered to the requester following Interest's reverse path, ensuing each intermediate router's information states. For this purpose, NDN routers are equipped with Pending Interest Table (PIT), which is mainly responsible for reverse-path forwarding. Given the IP addressing replacement, NDN resolves known Denial of Service (DoS) and Distributed Denial of Service (DDoS) issues—e.g., UDP, ICMP and SYN flood [4]. Nevertheless, NDN is not immune to possible flooding attacks. Here, a new type of DDoS – i.e., Interest Flooding Attacks (IFA) – represents the case of one or more compromised nodes issuing a high number of Interest requests. Thus, the attack exploits the router's PIT to congest the network and degrade or deny the legitimate users' service while consuming the router's resources.

Detection and mitigation of IFA have been the focus of several research proposals [5]–[7]. Generally, in these proposals, IFA is analysed in small topologies, and the detection mechanism is based on the local view of each router. In particular, the routers measure specific statistics that best describe IFA, and after that, the designed detection technique decides whether the specific router is under attack. Although valid, such detection schemes lack the network's global view, hindering the IFA detection in cumbersome scenarios. Conversely, from detection of well-known DDoS attacks in IP networks [8]–[10], Artificial Intelligence (AI) and Machine Learning (ML) algorithms have not been widely used for IFA detection. The lack of large and fully representative IFA datasets is the catalyst for such a lack. Motivated by the shreds of evidence mentioned above, we here collect a large IFA dataset. Afterward, we design an IFA detection technique based on Graph Neural Network (GNN) that maps the considered NDN network to a graph, representing the interactions among routers while overcoming the locality nature of state-of-the-art proposals. Overall, the contributions of this paper are the following:

- We collect the first dataset built for SPOTting IFA Intruders (SPOTIFAI)[1]. SPOTIFAI includes about 40 IFA scenarios constructed over three network topologies, corresponding to 21 hours of traffic. The scenarios used for collecting SPOTIFAI realistically mirror the IFA behaviour in NDN networks.
- We design GNN4IFA, an IFA detection technique based on GNNs representing NDN networks as graphs by mapping routers interconnections. GNN4IFA has a two-fold design – i.e., Supervised Attack Detection (SAD) and Unsupervised Attack Detection (UAD) – according to how we train the GNNs.
- We evaluate the performance of GNN4IFA in both configurations using SPOTIFAI.[2] Here, we train

1. SPOTIFAI is publicly available at https://tinyurl.com/7zp4z5zv
2. GNN4IFA source code is available at https://github.com/AndAgio/GNN_4_IFA

about 300 SAD and UAD models and study their performance. GNN4IFA reaches a detection rate higher than 99% and a negligible false positive rate while demanding short computation time. Additionally, we compare GNN4IFA with the state-of-the-art ML and non-ML based techniques, showing how GNN4IFA outperforms the implemented baselines by up to 6% accuracy, and 10% and 15% true positive rate, respectively.

- We analyse the generalizability of GNN4IFA detection on different topologies, showing its robustness and ease of deployment. To do so, we train SAD and UAD on one topology while testing them on the others. GNN4IFA represents the first IFA detection mechanism capable of generalising between topologies different from the one used in the design phase.

This article is organised as follows. In Section 2, we provide a brief background for ICN and GNN, while in Section 3, we describe the state-of-the-art proposals. Section 4 introduces the system and threat model and articulates the requirements for a robust, reliable, and generalisable IFA detection mechanism. Subsequently, Section 5 fully describes SPOTIFAI, from data collection to its setup. Section 6 and Section 7 present the design of GNN4IFA and its evaluation, respectively. Finally, we conclude and present the future directions in Section 9.

2. Background

In this section, we provide an overview of Information-Centric Networking (ICN) in Section 2.1, Interest Flooding Attacks (IFA) in Section 2.2 and Graph Neural Network (GNN) in Section 2.3.

2.1. Information-Centric Networking (ICN)

Since NDN is the most promising ICN architecture, we describe NDN and hereafter use the terms ICN and NDN interchangeably. Unlike host-based paradigms, content is the core abstraction of NDN [11]. With such a shift, NDN relies on functionalities such as content naming, name-based routing, caching, and name-based security [12]. Each content is recognized through a location-independent unique name similar to URLs and comprises one or more variable-length components. NDN communication follows the pull model — i.e., data is delivered only upon request. To accommodate name-based forwarding and caching features, NDN routers must include three components: *(i)* PIT that lists unsatisfied interests and their corresponding incoming interface; *(ii)* Content Store (CS) that caches the content; *(iii)* Forwarding Information Base (FIB) that lists the name prefixes and their corresponding outgoing interfaces. In the upstream path, the router first probes the CS using the name contained in the interest packet. In case of CS hit, data is immediately forwarded back to the requester. Otherwise, the router checks the PIT. If there are existing PIT entries, the router collapses all requests for the same content and adequately updates the list of incoming interfaces. Conversely, a new entry is added. In both cases, the FIB is consulted to forward the interest packet toward the next hop. Instead, in the downstream path, PIT is checked for a match. If PIT is hit, the router forwards the content to the corresponding interfaces and flushes the PIT entry. Elseway, the router drops the content packet. Lastly, the router stores the content in the CS following the caching policies. Thus, in the downstream path, the router follows the reverse routing considering only the PIT information.

2.2. Interest Flooding Attacks (IFA)

It takes a minimum effort for a NDN attacker to mount DoS or DDoS. Name-based routing consumes memory resources at intermediate routers, exposing PIT and CS components to potential DoS attacks. For example, an attacker (or a set of distributed compromised nodes) injects many requests for content to intentionally change the CS locality or overload the network while denying the service to legitimate consumers. Here, we focus on attacks that exploit the PIT, in particular, a saturation of the router's PIT through the rapid generation of interests, aiming to increase the drop rate for incoming interests and degrade or even deny the service for legitimate consumers. Such attacks are also known as Interest Flooding Attacks (IFA). Conversely from known TCP/IP DDoS attacks – e.g., SYN Flooding, ICMP Flooding or UDP Flooding [4] – that primarily target the servers, IFA targets the routers' resources. Based on the requested content, there are three ways to mount an IFA: *(i)* IFA for the existing namespace (static or dynamic); *(ii)* IFA for non-existing namespace; *(iii)* IFA for existing and non-existing namespaces. For the *(i)* case, the attacker continuously sends the interest packets for a set of existing contents that can be either static or dynamic [13]. Due to the presence of caches in the intermediate routers, the impact of the IFA launched with massive requests for static content is limited. Conversely, dynamically generated content affects legitimate consumers while consuming the resources of both routers and producers. Meanwhile, for the *(ii)* and *(iii)* cases, the attacker can also rely on sending interest requests for non-existing content. This article considers IFA with existing dynamic content.

2.3. Graph Neural Networks

GNNs have been recently proposed as Neural Network (NN) extensions to enable handling graph structured data. GNNs are mathematically defined to operate upon directed graphs, whose vertices (respectively, arcs) are labeled with real-valued vectors representing the information numerically at hand. We define $\mathbf{x}_v \in \mathbb{R}^d$ the vector carrying information concerning the vertex v, and $\mathbf{a}_{v,w} \in \mathbb{R}^c$ the vector depicting the arc between the vertex v and w—with c and d representing the dimensionality of the vector. GNNs rely on the *graph convolution* operation, defined as the generalisation of a 2-dimensional convolution over graph-structured data. The graph convolution is defined over a vertex v, and its neighbourhood $N(v)$ and applied simultaneously to each graph node G to update its representation. More in detail, the graph convolution relies on three successive phases:

Propagation. Here, each vertex v in G receives the information stored in its neighbours $\mathbf{x}_{v'} \, \forall v' \in N(v)$, weighted – via a function Ξ – by the information $\mathbf{a}_{v,v'}$ of the arc between v and v';

Aggregation. Here, the information received is aggregated via a predefined aggregation function $\uplus$;

Transformation. Here, the aggregated information is transformed – via a function Θ – into a new – more meaningful – embedding vector and assigned back to the vertex v as its new state $\mathbf{x}'_v$.

Selection of the *propagation*, *aggregation*, and *transformation* functions is made on hand and represents a design choice of the graph convolution.

GNNs have proven to be successful over many graph processing tasks, such as computational chemistry [14], social recommendations [15], computer vision [16], and many others [17], [18]. However, a comprehensive review of GNNs and the underlying techniques are clearly out of the scope of this paper: therefore, we refer interested readers to [19], [20].

3. Related Work

A series of statistical approaches have been first proposed to counteract IFA. Such mechanisms commonly rely on thresholding one or more metrics of interest. Depending on the thresholding operation, a set of nodes is selected to decide which interest requests to drop. Afanasayev et al. [5] propose one of the first threshold-based IFA detection on the routers' satisfaction ratio. After that, Poseidon [7] extends the analysis, also considering the PIT space usage while using dynamic thresholds. Also, Dai et al. [21] focus on PIT usage to identify malicious namespaces that consume PIT's entries. ChoKIFA [6] calculates three thresholds over the name and interest prefix match and satisfaction ratio. Later, ChoKIFA+ [22] proposed a detection mechanism based on the PIT consumption based on malicious interests. In addition, Xin et al. [23] thresholds over name prefix entropy since the occurrence of interest names changes significantly under IFA. Furthermore, Salah et al. [24], [25] propose to detect collusive IFA following double thresholding on PIT utilisation – i.e., locally and globally on a controller node. Wang et al. [26] propose the cooperative filter technique that leverages fuzzy logic and mitigates IFA based on router cooperation. Lastly, Benmoussa et al. [27] distinguish between intentional and unintentional behaviours due to network congestion. Therefore, the paper proposes a congestion-aware IFA detection and mitigation.

Although statistical approaches have exhibited promising results, they all lack generalisability and show restricted application potential. Moreover, the application of AI to IFA detection still needs to be explored, arguably due to the lack of meaningful IFA datasets. Zhi et al. [28] propose to leverage a Support Vector Machine (SVM) model for IFA detection, training the classifier over the entropy value of interest names, PIT usage, and interest satisfaction rate. The proposal was trained and tested in the small-scale binary tree topology via simulations. Two works by Kumar et al. [29], [30] analyse alternative ML approaches, training four classifiers over a set of selected features from collected IFA traffic. Both their proposals were trained and tested using a linear, small-scale binary tree and DFN topology. Chen et al. [31] proposed iForest, a detection and mitigation IFA mechanism constructed to detect malicious prefixes based on the PIT usage and the number of sent interests and received data packets. iForest is designed and tested only on the small-scale binary tree topology through simulations. Even though the results of these works seem promising, most lack transparency in the traffic collection phase, are evaluated in small-scale scenarios, and suffer the cost of identifying hand-crafted features.

Lastly, the interest in applying GNNs for detection on different tasks has gained attention in the cybersecurity research community. King et al. [32] proposed EULER, a dynamic link prediction mechanism employing a GNN model upon a Recurrent Neural Network (RNN) sequence encoder. Such application of GNN differs from our proposal as it deals with temporal link prediction. Additionally, Wang et al. [33] also leveraged GNNs for an anomaly detection mechanism. However, the proposed detection is for node-level threat detection, differing from our network-wide mechanism. Ji et al. [34] proposed NestedGNN, a compromised host detection through a nested GNN mechanism. NestedGNN differs from our mechanism as it considers a nested graph composed of three layers. Concluding, the common denominator of GNN4IFA and the above-mentioned works is the use of GNNs in distributed scenarios, reinforcing our claim of suitability of GNNs for IFA detection.

4. System and Threat Model

In this section we first describe the system model (Section 4.1) and the threat model (Section 4.2), then we elaborate on the requirements for our detection mechanism (Section 4.3).

4.1. System model

We consider a generic NDN network with a variable number of nodes—i.e., consumers, producers, and network devices. In particular, we consider three state-of-the-art network topologies: a small-scale binary tree topology; the DFN (German Research Network) topology [35]; the Rocketfuel topology [36]. We chose these topologies because of the heterogeneous network scenarios they represent. The small-scale topology is a bottleneck scenario where a single producer can satisfy all the content requests. Instead, DFN and Rocketfuel topologies have more nodes, thus closer to a real-world scenario. In our topologies, all producers handle requests for the same set of contents, which falls under a specific namespace.

4.2. Threat model

For the attack scenario, we consider the general purpose of the attacker to be the denial of access to a specific target set of contents—i.e., specific content namespace. To this end, the attacker can adopt one of the following techniques: *(i)* targeting an existing content namespace; *(ii)* useing non-existing content namespaces; *(iii)* shuffling both existing and non-existing namespaces. In our threat model the attacker targets existing content namespace by issuing many requests for dynamic content in the targeted namespace: we focus on this sort of IFAs as it represents the most challenging attack to detect. Unlike *(ii)* and *(iii)*, in our setup the attacker can try to hide her malicious behaviour by using valid existing namespaces. Instead, IFA relying on non-existing content namespaces can usually be detected by applying namespaces filtering techniques.

Different attacker capabilities can lead to IFA with varying impacts on the network. The attacker capabilities we consider are interest request frequency, number of compromised nodes, and location of compromised nodes in the network. The attacking frequency represents the most impactful IFA feature. Thus, we analyse IFA behaviour depending on these frequencies and identify the boundaries for an IFA attack to be considered ongoing. In practice, relatively low request frequencies used from the compromised nodes create traffic patterns akin to the normal traffic—i.e., no compromised nodes. By studying the effectiveness of IFA via the distribution of PIT sizes for various request frequencies, we aim to identify the most relevant attacking frequencies. Lastly, in the considered threat model, the compromised nodes share the same properties and are not considered to be coordinated, except for the start time of the attack. This attack setup represents the one that best fits a real-world scenario. The attacker has access to multiple devices, which, once activated, aim to overflow the network without further coordination.

4.3. Requirements

Designing a robust, precise, and generalisable IFA detection mechanism requires some criteria to be met:

C1 *Global approach.* The anatomy of DDoS attacks – including IFA – is founded on the interaction among compromised nodes distributed over the network. In this context, focusing only on local features detected in each network node emerges to be limited to capturing such interaction and, consequently, less accurate to detect IFA. Therefore, a more global approach would be preferable to increase detection precision while lowering false alarms.

C2 *Adaptable to heterogeneous topologies.* Generally, detection mechanisms are designed and tested in the same – usually small-scale – network topology, leading to topology-specific solutions. This hinders their deployment into real-world topologies different from those considered in the design phase. Thus, the mechanism should address its transferability over different topologies during the design phase.

C3 *Large and representative network traffic.* The common denominator of all the ML-based IFA detection mechanisms is the knowledge of the network traffic. Depending on the mechanism being supervised or unsupervised, a legitimate or malicious network traffic might be needed. In particular, supervised detection is mainly designed using labeled data representing legitimate and malicious network traffic. Instead, the unsupervised mechanisms demand legitimate network traffic only.

C4 *Robust and efficient detection.* The detection mechanism should be fast enough to identify an attack in real time. Furthermore, if the detection mechanism is allocated in the routers – which are the target of IFA – it is more likely to be devastated as soon as the attacker floods the network. Therefore, the detection mechanism should be wisely placed in the network.

Our proposal – detiailed in Section 6 – aims at incorporating all the criteria C1-C4. In particular, we use GNNs to enable our detection mechanism to extract information from the whole NDN network rather than focusing on specific features of single network devices, thus automatically satisfying C1. Meanwhile, relying on graph computation mechanisms, our approach shows generalisability characteristics missing from state-of-the-art approaches—see Sections 7.1 and 7.2. GNNs are proven to generalise well between different graph structures – especially concerning graph classification tasks [37] – and thus allow our approach to satisfy C2. Moreover, relying on GNNs to process graph data has also proven to be time efficient, as different approaches exist to optimise their application to huge-scale graphs [17], [38]. Therefore, our experiments show that our approach easily satisfies C4 even on large-scale NDN networks. Finally, to enable C3, we extract and propose the most comprehensive – up to our knowledge – dataset of IFAs in NDN networks—check Section 5. The proposed dataset spans different topologies and multiple attack setups, rendering the extracted network traffic representative of the most heterogeneous IFA attacks.

5. SPOTIFAI: SPOTting IFA Intruders

The availability of open source datasets and code reproducibility highly impact research related to IFA detection. To the best of our knowledge, no IFA datasets are available. Therefore, we collected SPOTIFAI by analysing several IFA scenarios using ndnSIM [39], an open-source platform that implements the NDN protocol. SPOTIFAI realistically mirror IFAs behaviour in NDN networks [35]. In Section 5.1, we describe the collected data and the attacking configuration parameters. Instead, in Section 5.2, we dive into the experimental details considered during the IFA implementation and analysis.

5.1. Data Collection

We assume the presence of a central node, assigned by the Internet Service Provider (ISP), to collect node traces and apply the IFA detection mechanism. We consider two configurations for collecting SPOTIFAI: *(i)* standard – i.e., if there are no compromised nodes in the network; *(ii)* under attack – i.e., if there is an ongoing attack. For the standard configuration, SPOTIFAI contains five simulations for each topology. Instead, for the configuration under attack, SPOTIFAI encompasses several scenarios for each topology, varying the number of compromised nodes and their attacking frequency. Due to the sessionless nature of NDN networks, collecting data following the traffic flow is not semantically representative. Therefore, we have considered 12 features for the analysis of IFA depicted in Table 1, among which ten are interface dependent. Concretely, a node with L interfaces collects $10 * L + 2$ features. We consider the traffic rate in each node's interface, including incoming and outgoing traffic. Here, we measure the incoming and outgoing rates for interest and data packets, satisfied interests, timed-out interests, and NACKs. Additionally, for each router, we consider the packet drop rate. Finally, we capture the PIT evolution by tracing the number of entries of PIT during each simulation. The considered features are selected as they compose the most comprehensive metrics describing each interface's router states. Additionally, most of the collected features are used in the state-of-the-art.

TABLE 1: Features description.

Feature Symbol	Description
f_1^l	# of received interests in an interface l
f_2^l	# of sent interests through an interface l
f_3^l	# of arrived data in an interface l
f_4^l	# of sent data through an interface l
f_5^l	# of satisfied interests per an incoming interface l
f_6^l	# of satisfied interests per an outgoing interface l
f_7^l	# of timed out interests per an incoming interface l
f_8^l	# of timed out interests per an outgoing interface l
f_9^l	# of NACKs per an incoming interface l
f_{10}^l	# of NACKs per an outgoing interface l
f_{11}	cumulative measurements of dropped packets per node
f_{12}	# of PIT entries per node

IFA severeness on the routers and service degradation strictly depend on the number of compromised nodes and their request frequency. Moreover, compromised nodes' location significantly affects the success of mounted IFA, especially when considering aggregation routers that connect multiple edge routers. Therefore, while collecting SPOTIFAI, we consider a variable number of compromised nodes with randomly selected locations and variable request frequency. Thus, we obtain a complete and thorough dataset representing the full spectrum of IFA effects on NDN networks.

5.2. SPOTIFAI Setup

IFA simulations are carried out using two widely used topologies in the state-of-the-art: small-scale binary tree topology and DFN (German Research Network). Conversely, from most existing proposals, we consider a more extensive topology—i.e., Rocketfuel topology [36]. We select these topologies to enable a fair comparison between existing works and our approach and extend it to a large-scale scenario—see Figure 1. The small-scale topology comprises 18 nodes, including 8 user nodes – i.e., both legitimate and compromised ones – and a single producer. The topology corresponds to one of the worst cases for an IFA detection mechanism where a single producer satisfies all users' requests. DFN topology is widely used by the state-of-the-art IFA detection proposals. It consists of 29 nodes with 12 user nodes – i.e., compromised and legitimate nodes – and six producers. The higher number of producers smoothes the IFA effect since users' requests are distributed across multiple producers. For small-scale and DFN topologies, we fix bandwidth and propagation delays to 10Mbps and 10ms, respectively. Instead, Rocketfuel – i.e., large-scale – comprises 163 nodes. Among these, 72 nodes are users, 68 gateways, and 23 backbone nodes. Here, the producer nodes are chosen from the set of gateways and backbones. We set variable bandwidth and propagation delay values according to the link types, as Afanasyev et al. [5] proposed.

To approximate a real network traffic behaviour in our experiments, we assume that legitimate users issue requests following the Zipf-Mandelbrot distribution [40]. Their request frequency is selected following a random uniform distribution in [50, 150] interests/second. Similarly, the compromised nodes follow the exact requests distribution pattern as legitimate users while using request frequencies that are multiples of legitimate users' frequencies. Similarly to other approaches [7], [30], we limit the maximum PIT capacity to 1200 entries. For every IFA simulation scenario setup, we perform five runs, each spanning five minutes. The attack starts in the 50s and stops at the end of the simulation. For each simulation, the statistics are collected every second. Table 2 summarizes the specific parameter values. Overall, SPOTIFAI accounts for 40 heterogeneous IFA attack scenarios and 250 collected runs. Therefore, SPOTIFAI contains about 21 hours of collected network traffic.

TABLE 2: Simulation parameters values.

Simulation Parameter	Value
Simulation duration	300s
Statistics collection time	1s
Maximum PIT size	1200
Legitimate users frequency	U∼[50-150] interests/s
Attackers frequency coefficient	[4x, 8x, 16x, 32x, 64x]
Attackers number	Small-scale: 4, 5, 6, 7
	DFN: 4, 8, 11
	Large: 15, 36, 64
Legitimate users runtime	0-300s
Attackers runtime	50-300s

Most existing proposals do not clearly define the precise criteria for an IFA to succeed in the scenarios they consider. We study the network behaviour under different combinations of compromised nodes' numbers and the frequency of their requests to overcome such limitations. Figure 2 depicts the PIT size – modeled as Gaussian distribution – for small-scale topology. Here, F refers to the attacking frequency, and N represents the number of compromised nodes. As expected and represented in the zoomed right-hand side plot, increasing compromised nodes' frequency – e.g., 8x (red curves), 16x (orange curves), 32x (pink curves), 64x (light blue curves) – leads to PIT size values which are distributed along the tail of the distribution. Thus, the routers' PITs are saturated for these frequencies, confirming the IFA effect. Conversely, following the left-hand side, for lower attacking frequencies – e.g., 4x (black curves) – the PIT size distribution is similar and overlaps with the legitimate distribution (green curve). Following the above findings, we consider the 4x attacking frequency to be similar to the legitimate scenario and therefore exclude it from IFA setup.

Lastly, to show the IFA effect, we analyse the PIT consumption on the routers considering a specific attacking frequency and number of compromised nodes. In Figure 3a, we illustrate the findings for the DFN topology, considering the frequency of the attack 32x and 8 attacking nodes. Due to space limitations, the reader can refer to https://tinyurl.com/2pd3k9tt for the small and large scale topology plots. IFA has different effects on the routers, depending on their location in the network. For example, Rout9 (brown line) oscillates at a 200 entries PIT occupancy, 2x less compared to Rout8 in the small-scale topology. Furthermore, we illustrate the consumer satisfaction rate – i.e., the rate of data packets received for the interest requests – in Figure 3b considering the

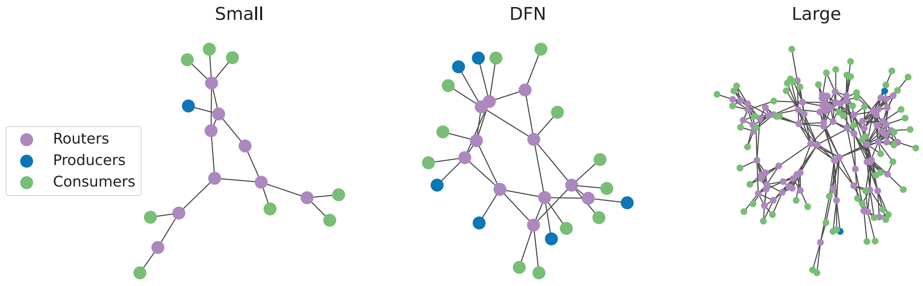


Figure 1: The three network topologies considered.

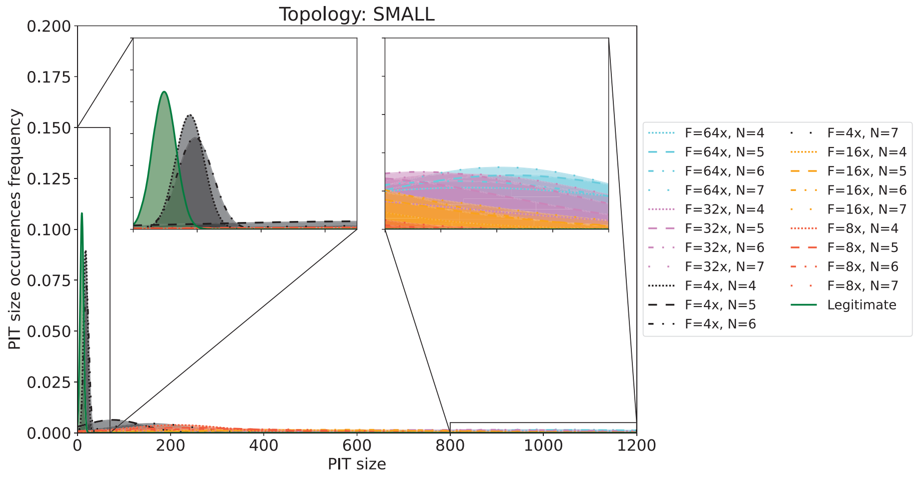


Figure 2: PIT size distribution for small-scale topology.

identical setups. As expected, IFA severeness in denying the service to the consumers is experienced as soon as the attack starts. Such effect is harsher for smaller topologies where IFA quickly floods all the routers. Besides Figures 2, 3a and 3b that show the behaviour of IFA in the SPOTIFAI, in Figure 3c we show its statistical stability in terms of standard deviation over time for the PIT size. In accordance with Figure 3a, Figure 3c shows the peak values of the standard deviation of PIT sizes as soon as IFA starts at time 50s. After that, a stable standard deviation is observed for all the attack duration, thus proving the simulation convergence.

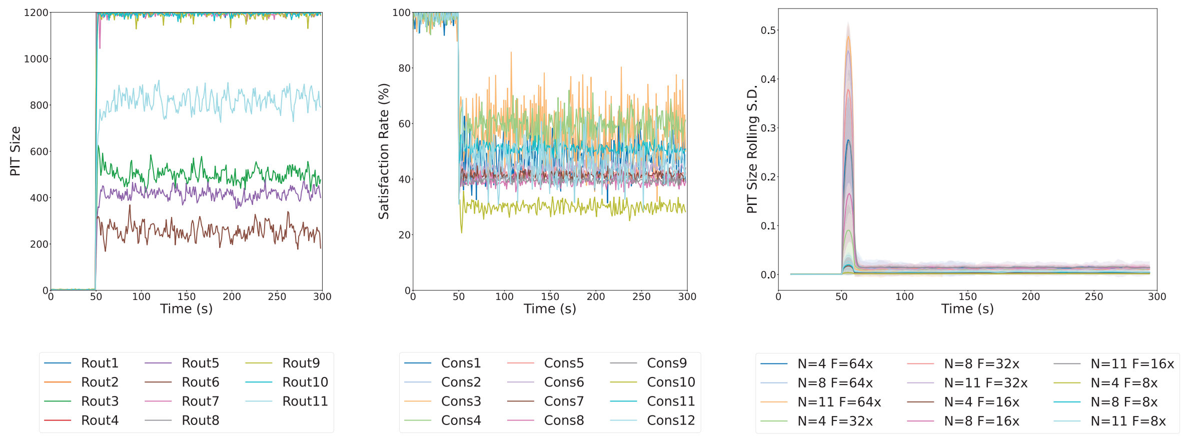
6. GNN4IFA

IFAs rely on compromised nodes' collaboration to inject an anomalous amount of requests, making them difficult to detect by standard security frameworks. Indeed, most – if not all – IFA detection mechanisms inspect local features to predict anomalous states. However, local features might remain unchanged during IFA. Given sufficient coordinated compromised nodes, it is possible to correctly mount an IFA by slightly increasing the number of requests they propagate while keeping the local behaviour – i.e., features – untouched. These simple yet sophisticated attacks will likely pass undetected by traditional detection frameworks. We here propose to leverage GNNs to identify IFA, given their ability to extract complex non-local information from graph-structured data. Indeed, GNNs extract the information concerning interaction among nodes of a graph and elaborate it to obtain the desired knowledge regarding the graph or nodes state. We propose to analyse an NDN network using GNNs to extract information about complex behavioural relations that may exist between multiple – malicious – nodes. We first provide an overview of the required steps to run GNN4IFA (Section 6.1), then we describe how the NDN traffic is mapped into a graph, representing connections among nodes (Section 6.2) . Lastly, we present the two GNN4IFA configurations (Sections 6.3 and 6.4).

6.1. GNN4IFA Overview

Here, we briefly introduce the GNN4IFA pipeline, consisting of 5 steps, as shown in Figure 4.

1) *Feature extraction from traffic*: Each router implementing GNN4IFA elaborates its traffic information to build the set of features necessary to detect IFA attacks (more details in Section 6.2).



(a) DFN topology: F=32x, N=8. (b) DFN topology: F=32x, N=8. (c) DFN topology, all setups

Figure 3: Effect of IFA – i.e., PIT consumption, consumer satisfaction rate, and PIT size rolling standard deviation – on the DFN topology.

2) *Features propagation to ISP*: The features extracted by each router are propagated to the ISP which is in charge of detecting the IFA.
3) *Construction of $\mathcal{G}(t)$*: Upon the reception of routers features, the ISP builds the graph $\mathcal{G}(t)$ representing the NDN network status. The construction of $\mathcal{G}(t)$ is described more in detail in Section 6.2.
4) *Run IFA detector*: $\mathcal{G}(t)$ is fed to the GNN detection mechanism that defines if an IFA is ongoing or not. This detection process can leverage both a supervised mechanism and an unsupervised approach and is described more in detail in Sections 6.3 and 6.4.
5) *IFA mitigation*: Upon attack detection, a mitigation scheme is deployed over the NDN network, aiming to reduce the IFA effects. There exist multiple approaches to mitigate IFAs upon their detection [21], [41], [42], and we here consider an NDN network relying on one of these mechanisms. Indeed, GNN4IFA is independent of the IFA mitigation as it can be coupled with any available scheme. Therefore, a detailed discussion concerning the deployed mitigation scheme is clearly out of the scope of this paper.

6.2. NDN Networks as Graphs

Before getting into details concerning the proposed GNN models, we must introduce how to map NDN networks into practical graphs to be fed to GNNs. The mapping from NDN scenarios to graph samples is not trivial, as multiple ways exist to define nodes and edges between them.

Given an NDN network at a specific time step t, we present how to map the network state into the graph $\mathcal{G}(t)$. We first focus on the topology of the NDN network at hand. Here, the interconnections among routers are considered to be known by the ISP running the detection mechanism. Generally, the ISP or the local network maintainer knows how routers are placed. Meanwhile, user nodes are discarded from the topology, given the privacy-preserving nature of NDN. Indeed, users cannot be univocally identified, and it is cumbersome to point out which users are connected to which router. Nevertheless, the lack of user nodes in $\mathcal{G}(t)$ does not represent a drawback. Indeed, router features are sufficient for detecting IFA since routers are the affected network components. Moreover, relying on user-node features may hinder attack detection since compromised nodes may act adversarially and try to modify their features to avoid being detected. The graph representing the NDN network is identified by $\mathcal{G}(t) = (\mathbf{X}(t), \mathbf{A}(t))$. Here, $\mathbf{A}^{N\times N}$ expresses the graph topology and is obtained by mapping interconnections between routers into a binary adjacency matrix. N represents the number of routers available in the NDN network, and the matrix entries are defined as:

$$a_{i,j}(t) = \begin{cases} 1, & \text{if } \mathit{link}(r_i, r_j, t) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $\mathit{link}(r_i, r_j, t)$ is a function that outputs 1 if a physical connection between router r_i and router r_j exists at time t, 0 otherwise.

The features representing the states of nodes in $\mathcal{G}(t)$ are obtained using the features f presented in Table 1. Since most of the selected features contain interface-specific information, we aggregate the features over a router's interfaces following Equation (2).

$$F_k = \sum_{l=1}^{L} f_k^l \quad \text{with } k = [1, 10] \tag{2}$$

Here, L represents the total number of interfaces for a router i, and f_k^l represents the kth feature of router i over its interface l. As f_{11} and f_{12} do not focus on router interfaces, Equation (2) is applied to features $k = 1$ up to $k = 10$. For seek of completeness, we then define $F_{11} = f_{11}$ and $F_{12} = f_{12}$. Therefore, for each router i, we obtain the set of features F_k with $k \in [1, 12]$ describing the router state aggregated over all its interfaces.

For each node i – i.e., router r_i – of the graph the features F are concatenated into a vector $\mathbf{x}_i(t) =$

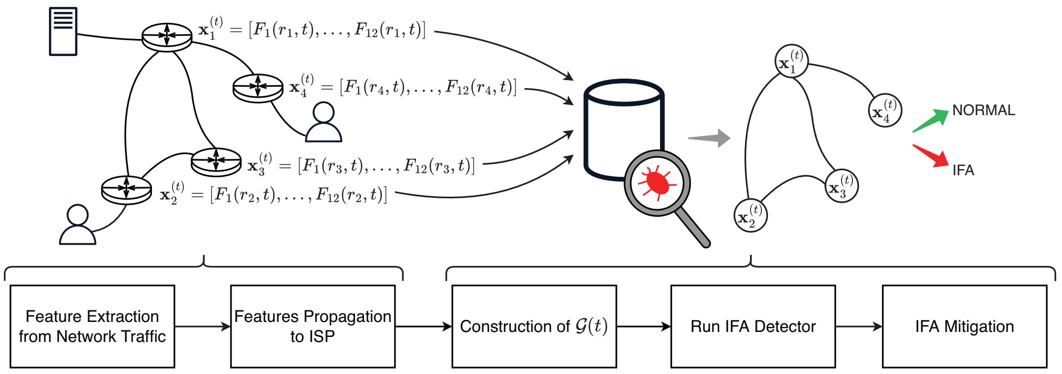

$\mathbf{x}_1^{(t)} = [F_1(r_1,t),\ldots,F_{12}(r_1,t)]$
$\mathbf{x}_4^{(t)} = [F_1(r_4,t),\ldots,F_{12}(r_4,t)]$
$\mathbf{x}_3^{(t)} = [F_1(r_3,t),\ldots,F_{12}(r_3,t)]$
$\mathbf{x}_2^{(t)} = [F_1(r_2,t),\ldots,F_{12}(r_2,t)]$
$\mathbf{x}_1^{(t)}$
$\mathbf{x}_4^{(t)}$
$\mathbf{x}_3^{(t)}$
$\mathbf{x}_2^{(t)}$
NORMAL
IFA
Feature Extraction from Network Traffic
Features Propagation to ISP
Construction of $\mathcal{G}(t)$
Run IFA Detector
IFA Mitigation

Figure 4: Overview of GNN4IFA. The routers of the NDN network collect features that characterise the traffic. These features are propagated to the ISP which is in charge of running the detection mechanism – SAD or UAD – and the attack mitigation.

$\left[F_1(r_i,t) \quad \cdots \quad F_{12}(r_i,t)\right]$. This vector represents the router state and is time-dependent, as it varies through time. The obtained vectors are then concatenated to obtain the matrix $\mathbf{X}^{N\times 12}$. Mathematically:

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}_1(t) \\ \vdots \\ \mathbf{x}_N(t) \end{bmatrix} = \begin{bmatrix} F_1(r_1,t) & \cdots & F_{12}(r_1,t) \\ F_1(r_2,t) & \cdots & F_{12}(r_2,t) \\ \vdots & \ddots & \vdots \\ F_1(r_N,t) & \cdots & F_{12}(r_N,t) \end{bmatrix} \tag{3}$$

Throughout our experiments, we consider a time window t of one second since SPOTIFAI is collected, keeping track of the features every second. Therefore, we build a graph $\mathcal{G}(t)$ for every second.

6.3. Supervised Attack Detection (SAD)

Given an input graph $\mathcal{G}(t)$ representing an NDN network state, the task of detecting if the network is under IFA can be easily mapped into a binary classification task—the most straightforward mapping between IFA detection and GNN training. In particular, we consider a graph classification task meant to predict a single label for the whole $\mathcal{G}(t)$ that describes whether the NDN network is under attack – i.e., label 1 – or not—i.e., label 0. We do not consider mapping IFA detection into a node classification task. Defining which routers are affected by the IFA and which ones are immune is not trivial.

To tackle the graph classification task, we build a simple GNN model relying on μ successive graph convolution layers, followed by an average-pooling layer that aggregates the feature vectors of the convoluted nodes—i.e., routers. The aggregated network representation is then passed to a fully-connected classification layer whose output represents our model prediction. The SAD GNN is then trained over the NDN network instances obtained from SPOTIFAI. Finally, since it is supervised, we emphasise that SAD requires the training set to contain samples of both normal NDN network state and attack state.

6.4. Unsupervised Attack Detection (UAD)

In conjunction with SAD, we here present UAD, aiming to obtain a GNN-based model capable of detecting IFA even when trained solely on samples of NDN normal states. To this end, UAD relies on self-supervision to learn to reconstruct masked nodes of a given input graph. More in detail, similarly to [43] and [44], given a graph representing the NDN network state, we mask a few nodes of the graph, obtaining a new nodes' features matrix $\mathbf{X}_{\mathcal{M}}$. We then pass the masked graph as input to the GNN model, which is in charge of reconstructing the original – unmasked – graph via prediction of the features of the masked nodes. During training, the GNN model is optimised to minimise the loss function:

$$\mathcal{L} = \mathcal{E}(\mathbf{X}_{\mathcal{M}}, \tilde{\mathbf{X}}_{\mathcal{M}}) \tag{4}$$

where, $\mathcal{E}$ represents a user-defined error function measuring the distance between the predicted nodes' values $\tilde{\mathbf{X}}$ over the mask $\mathcal{M}$ and their true values $\mathbf{X}$. Thus, the trained UAD model can correctly predict the legitimate node behaviour.

The main idea behind UAD is that the trained GNN model can reconstruct the masked graphs for legitimate NDN states but fails to predict correctly masked anomalous NDN graphs—i.e., graphs of NDN networks under IFA. Therefore, it is possible to identify IFA via imposing a threshold τ on the error between the predicted nodes' values $\tilde{\mathbf{X}}_{\mathcal{M}}$ and their true values $\mathbf{X}_{\mathcal{M}}$. Whenever the error overcomes τ, an anomaly is detected at the running time, and the model outputs an attack label. During training, the threshold τ is selected as the p-th percentile value of errors obtained over legitimate graph samples.

Rather than working solely on time step t, UAD receives in input the difference between the graph of the current time step and the previous one—i.e., $\mathcal{G}_{in} = \mathcal{G}(t) - \mathcal{G}(t-1)$. Here, the difference between the two graphs is computed as the difference between their feature matrices. This stands since IFAs modifies the network characteristics over a limited amount of time. Indeed, the amount of traffic in a network under attack increases rapidly before reaching a plateau, as shown in Figure 3a of Section 5.2. Therefore, an anomaly can be detected over the NDN network only during attack activation time. Indeed, anomalies cannot be identified over the plateaus reached a short time after the attack starts since no absolute knowledge of the attack behaviour is considered in UAD. Figure 5 shows UAD workflow.

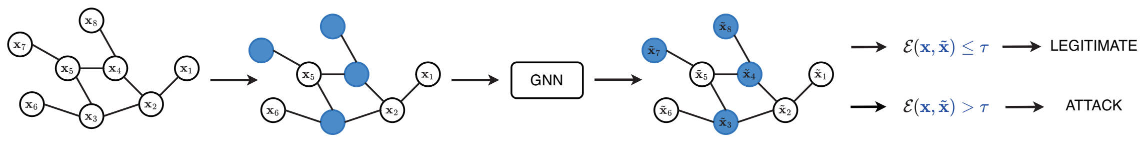

GNN
$\mathcal{E}(\mathbf{x},\tilde{\mathbf{x}}) \leq \tau$
LEGITIMATE
$\mathcal{E}(\mathbf{x},\tilde{\mathbf{x}}) > \tau$
ATTACK

Figure 5: The UAD workflow. The NDN network is converted into an equivalent graph. Randomly selected nodes are masked (blue vertices), and the GNN model aims at reconstructing their feature vectors. An attack is detected when the error score $\mathcal{E}$ between reconstructed and original vectors of masked nodes overcomes threshold τ.

7. Performance Evaluation

We here present the results obtained using SAD and UAD on SPOTIFAI in Section 7.1 and Section 7.2, respectively. In particular, we perform a detailed ablation study for each model, a thorough analysis of their detection ability, a comparison with the state-of-the-art, and an analysis of their detection generalisability, time requirements, and resource overhead.

7.1. Supervised Attack Detection (SAD)

7.1.1. Ablation study. Relying on graph convolution, SAD detection performance depends strongly on the selected convolution operation $\mathcal{T}$ and the number of convolutional layers μ. We investigate different combinations of $\mathcal{T}$ and μ to identify the best setup for SAD. More in detail, we sample $\mathcal{T} \in \{$GCN [45], Cheb [46], TAG [47], GIN [48], SG [49]$\}$, while letting μ span from 1 to 6. We train SAD over two runs of each SPOTIFAI setup – i.e., the combination of compromised nodes' request frequency and the number of compromised nodes – representing 40% of the SPOTIFAI and encompassing a total of 11920 samples. Among these, 2000 samples are benign; the remaining 9920 represent malicious behaviour. Instead, for testing SAD performance – i.e., accuracy and F1-score – we use the remaining three runs, aggregating the results. Overall, we use 17880 samples during testing, among which 3000 samples are benign, and the remaining 14880 represent an attack. Throughout the remainder of SAD experiments, we maintain the same data proportions for training and testing.

Figure 6 shows the results of our ablation study over the small and DFN topologies. The best hyperparameters setup for SAD is represented by the combination $\mathcal{T}$ = GIN and $\mu = 2$. This model reaches more than 98% IFA detection accuracy and F1-score for both small and DFN topologies. Nevertheless, our study shows that precise detection can be obtained using all the graph convolution operations analysed. On the flip side, it is interesting to notice that higher values of μ lead to performance degradation, possibly due to overfitting issues.

7.1.2. Performance analysis. SAD detection performance above shows an aggregate estimation, where metrics are averaged over all testing runs. Table 3 shows the accuracy and F1-score obtained by the best SAD model – i.e., $\mathcal{T}$ = GIN and $\mu = 2$ – on all number of compromised nodes and their interest frequency in the DFN and large topology. Our model successfully detects IFA for all scenarios, reaching detection accuracy and F1-score higher than 98% and 99%, respectively. Meanwhile, in the small-scale topology, our model reaches 100% accuracy and F1-score over all the setups. As expected and shown in Table 3, SAD presents higher detection performance when dealing with compromised nodes setup where IFA are more severe—i.e., higher request frequency and higher number of compromised nodes. Moreover, SAD performs slightly worse on the DFN topology. Indeed, such a topology is flooding resistant, lacking communication bottlenecks. Lastly, we tested the SAD's robustness by scaling the percentage of train samples. The results shown in Table 4 validate the strength of SAD, highlighting small performance drop over the large topology when training with only 5% of SPOTIFAI samples, corresponding to 1192 samples. Meanwhile, over smaller topologies SAD shows no degradation in performance. In conclusion, the results show SAD's reliability for detecting IFA in all setups.

7.1.3. Comparison with ML baselines. We tested our SAD mechanism with four state-of-the-art baselines that use ML algorithms for IFA detection [28]–[31]. Here, two baselines [29], [30] use multiple ML algorithms. Therefore we compare GNN4IFA to all the available ML combinations. *Due to the lack of publicly available datasets and source codes used by the baselines, reproducing them becomes a non-trivial task.* We replicated the proposed detection mechanisms by relying on the SPOTIFAI dataset for extracting the set of features used by each baseline that are available in the dataset. After that, the routers' features at each time step are embedded into a one-dimensional vector – via concatenation – that is used for IFA classification. Here, we split the available data into 20% used for training the baseline, 40% for its validation, and the remaining 40% for its testing. Table 5 shows the accuracy and F-1 score of the baselines and GNN4IFA in the three considered topologies. GNN4IFA SAD achieves identical or comparable results to the baselines for the small and DFN topology. Instead, for the large topology, it outperforms the baselines achieving at least a 6% of improvement in accuracy and a 5% of F-1 score. Indeed, GNN4IFA reaps the benefits of the knowledge gained from the large topologies, where it can explore the complex interaction among compromised nodes during an IFA. Conversely, the baselines focus only on locally measured features on the routers while completely lacking the wide network view.

7.1.4. Comparison with non-ML baselines. We compare our SAD mechanism with five state-of-the-art non-ML

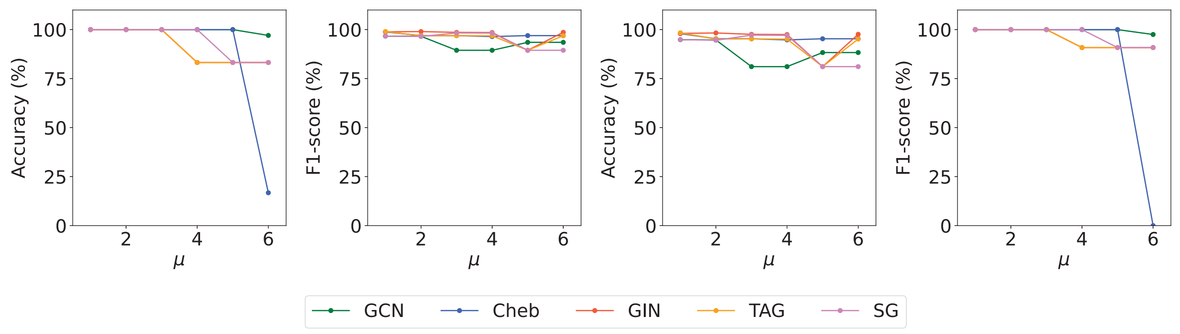


Figure 6: SAD ablation study. Accuracy and F1-score are shown concerning the μ and $\mathcal{T}$ (see Section 6.3). Left side plots refer to the small topology, while the right side refers to DFN.

TABLE 3: Accuracy (top) and F1-score (bottom) of SAD over each SPOTIFAI setup for DFN (left) and large (right) topology. Average score are shown in the last column and row of the tables.

N \ F	8×	16×	32×	64×	Avg.
4	92.1% 95.5%	95.6% 97.5%	97.0% 98.2%	95.0% 97.1%	94.9% 97.0%
7	100.0% 100.0%	99.9% 99.9%	100.0% 100.00%	100.0% 100.0%	100.0% 100.0%
11	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%
Avg.	97.4% 98.5%	98.5% 99.1%	99.0% 99.4%	98.3% 99.0%	**98.3%** **99.0%**

N \ F	8×	16×	32×	64×	Avg.
15	93.1% 95.2%	100.0% 100%	99.5% 99.7%	100.0% 100.0%	98.4% 98.7%
16	100.0% 100.0%	100.0% 100.0%	100.0% 100.00%	100.0% 100.0%	100.0% 100.0%
65	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%
Avg.	97.9% 98.4%	100.0% 100.0%	99.8% 99.9%	100.0% 100.0%	**99.5%** **99.6%**

TABLE 4: Accuracy (top) and F1-score (bottom) of SAD over SPOTIFAI topologies, when trained on a different percentage of samples.

Topology	60%	40%	20%	10%	5%
Small	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%
DFN	98.5% 99.1%	98.6% 99.2%	98.5% 99.1%	98.5% 99.2%	98.4% 99.0%
Large	99.5% 99.6%	99.4% 99.5%	98.3% 98.8%	98.0% 98.5%	94.8% 96.3%

baselines [6], [7], [24], [26], [27] using the SPOTIFAI dataset. Given the locality of the baselines, for a fair comparison, we calculate from SPOTIFAI the underlying features for each router's interface, as shown in Table 6. Each router provides a prediction based on the feature vectors at a specific time step. After that, the predictions from all the routers are aggregated into a one-dimensional vector. To establish a reasonable comparison with our global SAD mechanism, we consider an attack ongoing in the case at least one of the routers predicts IFA. Table 6 shows the accuracy and F-1 score for each baseline on the three considered topologies. GNN4IFA SAD achieves sharply better accuracy and F-1 score results, confirming the global detection mechanism's superiority over local detection.

7.1.5. Generalisability. We aim to test the ability of SAD to generalise its detection performance to unseen topologies. To the best of our knowledge, this work represents the first approach that studies generalisability aspects of IFA detection. This component is fundamental as it shows if and to what extent a detection mechanism can be deployed in real-world scenarios. Indeed, in concreteness, a detection system is expected to be deployable over multiple heterogeneous networks characterised by different structures. Given the supervised nature and the lack of transferability study of the baselines, we present only the generalisability results of GNN4IFA SAD. To this end, we train the SAD model on each topology, obtaining a topology-specific detector. Such a model is then tested on the other three topologies, computing the aggregated detection accuracy and F1 score. Table 7 shows the obtained performance. SAD can generalise from complex to simple scenarios—e.g., from large to the small topology. On the flip side, due to its supervised nature, SAD obtains degraded performance when transferred to larger topologies—e.g., from DFN to large. In particular, SAD struggles to detect IFA for the lowest values of compromised nodes request frequency—e.g., 8×. Indeed, in such scenarios, the network features may resemble legitimate behaviour and require specific detection knowledge. Such specific knowledge is not obtained when trained on a smaller topology, where IFA are more severe. The obtained behaviour is expected, as generalisability limitations represent well-known issues of supervised approaches. Overall, the results are encouraging, as they show how GNN4IFA represents the first detection system capable of correctly identifying IFA when deployed on a network that differs from the training one.

7.1.6. Time performance and resource overhead. Computation time requirements are relevant when dealing with

TABLE 5: SAD comparison with the baselines. Accuracy (top) and F1-score (bottom) measured on SPOTIFAI using selected features previously presented in Equation (2). Legend: MLP - Multi Layer Perceptron, SVM - Support Vector Machine, RF - Random Forests, DT - Decision Tree, NB - Naïve Bayes.

Baseline	Features	Topologies		
		Small	DFN	Large
[29] (MLP)	$[F_1{:}F_6,\ F_{12}]$	100.0%	100.0%	93.6%
		100.0%	100.0%	94.4%
[29] (SVM)	$[F_1{:}F_6,\ F_{12}]$	100.0%	100.0%	93.5%
		100.0%	100.0%	94.3%
[28] (SVM)	$[F_{13} = \frac{F_3}{F_2},$	100.0%	95.2%	93.5%
	$F_{14} = \frac{F_{12}}{1200}]$	100.0%	97.1%	94.4%
[31] (RF)	$[F_2,\ F_3,$	100.0%	100.0%	93.5%
	$F_8,\ F_{12}]$	100.0%	100.0%	94.4%
[30] (DT)	$[F_1{:}F_8,$	100.0%	99.9%	78.6%
	$F_{11},\ F_{12}]$	100.0%	99.9%	78.6%
[30] (NB)	$[F_1{:}F_8,$	83.2%	82.3%	60.7%
	$F_{11},\ F_{12}]$	90.8%	90.3%	75.6%
[30] (SVM)	$[F_1{:}F_8,$	100.0%	100.0%	93.5%
	$F_{11},\ F_{12}]$	100.0%	100.0%	94.4%
[30] (MLP)	$[F_1{:}F_8,$	100.0%	100.0%	93.5%
	$F_{11},\ F_{12}]$	100.0%	100.0%	94.4%
SAD (GIN)	**All**	**100.0%**	**98.3%**	**99.5%**
		100.0%	**99.0%**	**99.6%**

TABLE 6: SAD comparison with the non ML baselines. Accuracy (top) and F1-score (bottom) measured on SPOTIFAI using selected features presented in Table 1.

Baseline	Features	Topologies		
		Small	DFN	Large
[7]	$[\frac{f_1^l}{f_4^l},\ \frac{f_{12}^l}{f_1^l}]$	99.0%	79.9%	93.9%
		99.4%	86.1%	95.3%
[26]	$[\frac{f_{12}^l}{1200},\ \frac{f_7^l}{f_7^l+f_5^l}]$	75.9%	60.7%	94.4%
		83.0%	68.7%	95.4%
[24]	$[\frac{f_{12}^l}{1200},\ \frac{f_7^l}{f_7^l+f_5^l}]$	66.5%	81.0%	91.8%
		74.8%	87.0%	93.5%
[27]	$[\frac{f_3^l}{f_1^l},\ f_1^l,\ f_7^l,\ f_9^l]$	83.2%	82.3%	61.4%
		90.8%	90.2%	76.1%
[6]	$[\frac{f_1^l}{f_4^l},\ \frac{f_{12}^l}{f_1^l}]$	88.3%	43.3%	88.9%
		92.4%	47.5%	90.1%
GNN4IFA	**All**	**100.0%**	**98.3%**	**99.5%**
		100.0%	**99.0%**	**99.6%**

IFA detection. Indeed, lagged predictions lead to undetected IFA, causing network malfunction. To test the SAD time performance, we keep track of its inference time over a set of 14 thousand samples. We run SAD on a consumer laptop – i.e., Apple M1 chip, 16 GB RAM – obtaining an average prediction time of 0.351 ± 0.005 ms. Given this performance, it is safe to state that SAD can be deployed in real-world scenarios to promptly detect IFA, thus, protecting NDN networks from IFA at their very early stages. We run more experiments to measure the CPU and RAM usage for inference and training of GNN4IFA. Inference over 17880 samples requires a peak memory usage of 384 MBs and a peak CPU usage of 100% over two Apple M1 chip (octa-core) cores—the average CPU usage while testing is only 20%. Meanwhile, the model training process requires peak memory usage of 349 MBs and 100% CPU usage over a single core. SAD retraining requires, on average, 180 seconds over a set of 11920 samples. The obtained results show lower overhead compared to other solutions [50].

TABLE 7: Accuracy and F1-score of SAD over SPOTIFAI topologies, when trained on a different topology.

Topology transfer	Accuracy	F1-score
Small $\rightarrow$ Small	100.0%	100.0%
Small $\rightarrow$ DFN	72.8%	68.0%
Small $\rightarrow$ Large	72.8%	67.3%
DFN $\rightarrow$ Small	93.2%	96.1%
DFN $\rightarrow$ DFN	98.3%	99.0%
DFN $\rightarrow$ Large	54.9%	32.4%
Large $\rightarrow$ Small	100.0%	100.0%
Large $\rightarrow$ DFN	94.0%	94.2%
Large $\rightarrow$ Large	99.5%	99.6%

7.2. Unsupervised Attack Detection (UAD)

7.2.1. Ablation study. Similarly to Section 7.1.1, we aim to identify the best hyper-parameters for UAD. Differently, from SAD, the most influential parameter is represented by the p-th percentile used to identify threshold τ. Therefore, we here investigate different combinations of graph convolution $\mathcal{T}$ and p, showing their respective True Positive Rate (TPR) and False Positive Rate (FPR). UAD is trained over benign samples only, counting 891 samples in our experiments, and is tested over 22524 samples built as a mixture of benign and attack samples. Since UAD relies on time difference (see Section 6.4), we here consider a UAD anomaly to be a true positive if raised after the attack start time. Conversely, it is a false positive if raised before the attack starts. The UAD model uses 2μ graph convolutions. The former μ layers work as encoders while the latter as decoders. We set $\mu = 2$, representing the best value obtained for SAD. Moreover, we set the mask sampling probability to 0.3—i.e., randomly sample 30% of the graph's nodes. Variations on the masking probability do not influence much UAD's detection performance. Indeed, IFA produce substantial heterogeneity in nodes features, making the selection of the nodes to be masked indifferent. Finally, we use a Mean Squared Error measure for $\mathcal{E}$. Figure 7 shows the results of this ablation study over the small and DFN topologies. UAD best setup is obtained with $\mathcal{T} = GIN$ and $p = 0.985$. Indeed, such a model reaches 100% TPR and FPR smaller than 2% for both topologies. As expected, smaller p values lead to more frequent alarms, as the underlying anomaly detector is more sensitive to slight variations. This leads to higher TPR and FPR. Meanwhile, higher p values lead to a more robust anomaly detector, which avoids raising unjustified alarms – i.e., small FPR. However, higher p may also lead to degradation of TPR, which may cause undetected IFA.

7.2.2. Performance analysis. Similarly to Section 7.1.2, we present a microscopic analysis of UAD detection performance. Tables 8 to 10 present the TPR and FPR obtained by the selected UAD model – i.e., $\mathcal{T} = GIN$ and $p = 0.985$ – for each attack setup—i.e., number of

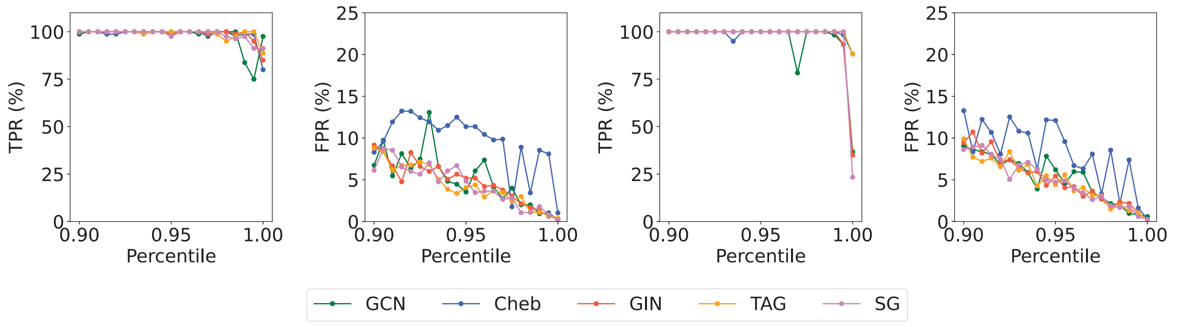


Figure 7: UAD ablation study. TPR and FPR are shown with respect to the p-th percentile used to identify threshold τ (see Section 6.4). Left side plots refer to the small topology, while the right side refers to DFN.

TABLE 8: TPR (top) and FPR (bottom) of UAD over each SPOTIFAI setup for the small topology. Average scores are shown in the last column and row of the tables.

N \ F	8×	16×	32×	64×	Avg.
4	100.0% 2.0%	100.0% 0.8%	100.0% 1.2%	100.0% 0.8%	100.0% 1.2%
5	100.0% 0.8%	100.0% 2.0%	100.0% 3.6%	100.0% 0.8%	100.0% 1.8%
6	100.0% 1.6%	100.0% 4.8%	100.0% 2.4%	100.0% 2.4%	100.0% 2.8%
7	100.0% 2.4%	100.0% 2.8%	100.0% 2.8%	100.0% 2.0%	100.0% 2.5%
Avg.	100.0% 1.7%	100.0% 2.6%	100.0% 2.5%	100.0% 1.5%	**100.0%** **2.1%**

compromised nodes and their interest frequency. UAD detects successfully IFA for all scenarios, even if trained solely on normal status—i.e., network not under IFA. In particular, our model reaches 100% TPR for each setup while keeping FPR under control—i.e., around 2% for most setups. *Here, we recall that GNN4IFA does not produce one inference per packet – as many other state-of-the-art mechanisms –, but rather relies on a time-step t inference. Indeed, SAD and UAD receive in input a graph $\mathcal{G}(t)$ – corresponding to the network state over the time window t – and produce one inference over it. Thus, GNN4IFA runs once every t seconds, and its corresponding TPR and FPR should be evaluated, considering this. Concretely, an FPR of 2% would result in 2 false alarms every 1000 seconds when setting $t = 10$, rather than resulting in 20 thousand false alarms per seconds over a network traffic of 1 million packets per second.* Lastly, we tested the UAD's robustness by scaling the percentage of train samples and cross-validating. Table 11 shows the results confirming UAD robustness, even when training only on 10% of SPOTIFAI corresponding to only 89 training samples—recall that UAD trains on legitimate samples only.

7.2.3. Comparison with ML baselines. To the best of our knowledge, all state-of-the-art IFA detection mechanisms rely on supervised ML algorithms. Therefore, for UAD comparison, we consider the supervised baselines –

TABLE 9: TPR (top) and FPR (bottom) of UAD over each SPOTIFAI setup for the DFN topology. Average scores are shown in the last column and row of the tables.

N \ F	8×	16×	32×	64×	Avg.
4	100.0% 2.4%	100.0% 3.2%	100.0% 2.0%	100.0% 2.0%	100.0% 2.4%
8	100.0% 0.8%	100.0% 2.0%	100.0% 2.0%	100.0% 1.6%	100.0% 1.6%
11	100.0% 1.6%	100.0% 1.2%	100.0% 1.6%	100.0% 1.2%	100.0% 1.4%
Avg.	100.0% 1.6%	100.0% 2.1%	100.0% 1.8%	100, 0% 1.6%	**100.0%** **1.8%**

TABLE 10: TPR (top) and FPR (bottom) of UAD over each SPOTIFAI setup for the large topology. Average scores are shown in the last column and row of the tables.

N \ F	8×	16×	32×	64×	Avg.
15	100.0% 1.0%	100.0% 0.0%	100.0% 1.0%	100.0% 0.0%	100.0% 0.5%
36	100.0% 0.0%	100.0% 3.0%	100.0% 1.0%	100.0% 2.0%	100.0% 1.5%
65	100.0% 0.0%	100.0% 0.0%	100.0% 1.0%	100.0% 0.0%	100.0% 0.25%
Avg.	100.0% 0.3%	100.0% 1.0%	100.0% 1.0%	100, 0% 0.6%	**100.0%** **0.75%**

i.e., [28]–[31] – and measure their TPR and FPR. Table 12 shows the results for three topologies. The UAD mechanism of GNN4IFA obtains a TPR identical to the baselines for the small and DFN topology. Meanwhile, for the large topology, it outperforms the baselines achieving at least a 10% increase in TPR. Such results confirm the benefits of leveraging topology information for detecting IFA on large topologies. Conversely, in topologies with few nodes – i.e., small and DFN – the topology information is not strongly relevant. Thus, detection with GNN achieves performance similar to the state-of-the-art.

Concerning the FPR, our UAD mechanism maintains it around 2% for the small and DFN topology, while it can be considered negligible for the large. Since it is trained on unlabelled data, higher FPR values are generally expected. Indeed, training UAD receives no information concerning

TABLE 11: TPR (top) and FPR (bottom) of UAD over SPOTIFAI topologies, when trained on a different percentage of samples.

Topology	90%	70%	50%	25%	10%
Small	100.0% 0.63%	100.0% 1.52%	98.75% 2.97%	98.75% 3.21%	97.5% 1.45%
DFN	100.0% 1.90%	100.0% 1.4%	100.0% 1.97%	100.0% 1.70%	100.0% 1.47%
Large	100.0% 0.73%	100.0% 0.68%	100.0% 0.66%	100.0% 0.51%	100.0% 1.16%

TABLE 12: UAD comparison with the baselines. TPR (top) and FPR (bottom) measured on SPOTIFAI using selected features previously described in Table 1.

Baseline	Features	Topologies		
	Small	DFN	Large	
[29] (MLP)	$[F_1{:}F_6,\, F_{12}]$	100.0% 0.0%	100.0% 0.0%	89.4% 0.0%
[29] (SVM)	$[F_1{:}F_6,\, F_{12}]$	100.0% 0.0%	100.0% 0.0%	89.3% 0.0%
[28] (SVM)	$[F_{13} = \frac{F_3}{F_2},$ $F_{14} = \frac{F_{12}}{1200}]$	100.0% 0.0%	96.9% 12.4%	89.4% 0.0%
[31] (RF)	$[F_2,\, F_3,$ $F_8,\, F_{12}]$	100.0% 0.0%	100.0% 0.0%	89.4% 0.0%
[30] (DT)	$[F_1{:}F_8,$ $F_{11},\, F_{12}]$	100.0% 0.0%	99.9% 0.0%	64.7% 0.0%
[30] (NB)	$[F_1{:}F_8,$ $F_{11},\, F_{12}]$	99.9% 100.0%	100.0% 100.0%	100.0% 100.0%
[30] (SVM)	$[F_1{:}F_8,$ $F_{11},\, F_{12}]$	100.0% 0.0%	100.0% 0.0%	89.3% 0.0%
[30] (MLP)	$[F_1{:}F_8,$ $F_{11},\, F_{12}]$	100.0% 0.0%	100.0% 0.0%	89.4% 0.0%
UAD (GIN)	**All**	**100.0%** **2.1%**	**100.0%** **1.8%**	**100.0%** **0.75%**

the difference between normal and attack traffic, which is valuable information that is used during the training of the baseline counterparts. Meanwhile, for the more extensive topology, UAD yields a more robust approach in terms of attack detection at the cost of some acceptable false alarms. Here, the best baseline – i.e., [29] –misses IFA about 10% of the times. In a real-world scenario, continuously detecting an attack while having some false alarms would be preferable compared to missing some of the attacks to have $FPR = 0\%$.

7.2.4. Comparison with non-ML baselines. Here, we compare our UAD mechanism of GNN4IFA with the non-ML baselines, measuring their TPR and FPR as shown in Table 13. UAD outperforms the baselines in the three topologies in terms of TPR. Instead, regarding the FPR, our GNN4IFA UAD outperforms all the baselines for the large topology, confirming the benefits of the global knowledge in such scenarios. On the other hand, for smaller topologies – i.e., small and DFN – UAD reaches higher, yet negligible values of FPR. Again, such a result is strongly related to the need for more relevant topology information in scenarios with limited nodes.

TABLE 13: UAD comparison with non ML baselines. TPR (top) and FPR (bottom) measured on SPOTIFAI using selected features previously presented in Table 1.

Baseline	Features	Topologies		
		Small	DFN	Large
[7]	$[\frac{f_1^l}{f_4^l},\, \frac{f_{12}^l}{f_1^l}]$	98.9% 0.0%	75.6% 0.0%	99.9% 15.5%
[26]	$[\frac{f_{12}^l}{1200},\, \frac{f_7^l}{f_7^l+f_5^l}]$	71.0% 0.0%	52.3% 0.0%	94.8% 6.2%
[24]	$[\frac{f_{12}^l}{1200},\, \frac{f_7^l}{f_7^l+f_5^l}]$	59.7% 0.0%	77.0% 0.0%	95.7% 14.2%
[27]	$[\frac{f_3^l}{f_1^l},\, f_1^l, f_7^l, f_9^l]$	100.0% 100.0%	100.0% 100.0%	100.0% 100.0%
[6]	$[\frac{f_1^l}{f_4^l},\, \frac{f_{12}^l}{f_1^l}]$	86.0% 0.0%	31.1% 0.0%	82.0% 0.0%
GNN4IFA	**All**	**100.0%** **2.1%**	**100.0%** **1.8%**	**100.0%** **0.75%**

TABLE 14: TPR and FPR of UAD over SPOTIFAI topologies, when trained on a different topology. Values with $*$ and $\dagger$ correspond to *p=0.61* and *p=0.995*, respectively.

Topology transfer	TPR	FPR
Small $\rightarrow$ Small	100.0%	2.1%
Small $\rightarrow$ DFN	98.3%	0.7%
Small $\rightarrow$ Large	91.7%*	0.0%*
DFN $\rightarrow$ Small	98.8%	2.2%
DFN $\rightarrow$ DFN	100.0%	1.8%
DFN $\rightarrow$ Large	100.0%†	2.9%†
Large $\rightarrow$ Small	100%	35.4%
Large $\rightarrow$ DFN	100%	26.9%
Large $\rightarrow$ Large	100.0%	0.75%

7.2.5. Generalisability. Differently from SAD, UAD can generalise from smaller to larger topologies, as shown in Table 14. Indeed, we obtain a minimal performance degradation when testing UAD on larger topologies. UAD's ability to generalise can be attributed to its unsupervised nature. Indeed, this approach learns solely to represent legitimate network behaviors, similar to most topologies. Meanwhile, attacks are detected as deviations from legitimate behaviours that are easily generalisable. However, UAD needs more generalisation from large to simpler topologies. The result stems from the high number of requests seen during training on the large topology due to the higher number of legitimate and compromised nodes. The results highlight the superiority of UAD over SAD and encourage its deployment in real-world scenarios.

7.2.6. Time performance and resource overhead. UAD is tested on the same consumer laptop of Section 7.1.6, where it obtains an average prediction time of 0.548 ± 0.004 ms over 22 thousand samples. Interestingly, UAD is 1.5 times slower than SAD, as it uses twice the convolutional layers μ. Moreover, UAD relies on node features reconstruction over the mask $\mathcal{M}$ and error $\mathcal{E}$ computation. Therefore, UAD needs to produce N inferences, representing the number of vertices in $\mathcal{G}$, rather than only a binary prediction. Given this performance, UAD can compute up to 4 thousand predictions per second, enabling its deployment in real-world scenarios. Additionally, we measure the CPU and RAM usage for inference and training of UAD. Inference over 29700 samples requires a peak memory usage of 671 MBs, and the average CPU usage while testing is only 25.4%. Meanwhile, the model training process requires peak memory usage of 656 MBs and 100% CPU usage over a single core. UAD retraining requires, on average, 8.23 seconds over 891 samples.

8. Limitations

The proposed GNN4IFA mechanism faces certain limitations mainly related to its detection rationale and the leveraging of AI algorithms for the detection. Lastly, the collected SPOTIFAI dataset may be subject to potential biases since it is collected in a simulated environment: so, we elaborate more on the aforementioned limitations.

GNN4IFA weaknesses. Designing a global IFA detection mechanism is challenging as it requires coordination among all the network nodes. Our GNN4IFA is purposely designed based on the GNN that maps the network to a graph for a global detection approach. The rationale behind GNN4IFA classifies the whole graph as under an IFA or not instead of considering per-router classification. While approaching this rationale, our GNN4IFA overcomes the locality issue of current IFA detection mechanisms, it introduces a few limitations. Here, if GNN4IFA detects an attack, the mitigation mechanism is implied to be activated on all the routers. However, one could argue that GNN4IFA can be coupled with proactive mitigation techniques which limit the routers' traffic depending on how severely the attack affects them. Moreover, leveraging state-of-the-art eXplainable Artificial Intelligence (xAI) techniques on GNN [51], we could easily identify the most affected nodes as the most important ones for attack detection and activate the mitigation only on those. Another limitation related to the GNN4IFA rationale includes the propagation of collected traffic toward the central entity in charge of deploying the detection mechanism. Such an issue can be tackled by relying on ad-hoc overlay – or underlay – network [10] or embedding the propagation in the NDN network traffic.

Lastly, GNN4IFA might be susceptible to adversarial attacks, mainly exploiting the graph structure [52]. Nevertheless, these attacks are mainly successful in white-box and grey-box scenarios, where the attackers know the model, the used features, and the labels. Additionally, mimicry attacks that modify the attack to mimic normal traffic patterns might disturb the detection robustness of GNN4IFA. Here, the SAD is bounded by its supervised nature. Therefore, changes in the attack patterns perturb its detection accuracy since attacking traffic will pass undetected. On the other hand, we expect UAD to be more robust from this perspective since it is trained on benign traffic. Therefore, the mechanism will detect even slighter changes in the benign traffic pattern.

SPOTIFAI biases. Collecting the SPOTIFAI, we considered state-of-the-art IFA simulations where the benign traffic is usually modeled following a Zipf-Mandelbrot distribution for consumer requests considering a fixed requesting frequency. For SPOTIFAI, we approached the same distribution but changed the requesting frequency of each consumer in each run. Such an approach is a widely used set up in simulation-based works [53], [54], reassembling faithfully real-world scenarios. However, SPOTIFAI is bounded to the simulation environment where it is collected, limiting the collection of benign samples from real-world dynamic traffic.

9. Conclusions and Future Works

ICN, and consequently NDN, as a new and not yet deployed technology has unsolved DoS and DDoS issues. This paper presents a two-fold contribution in hindering IFA, a severe DDoS issue where the attacker exploits routers' resources to degrade – or even deny – the service to legitimate users. Initially, we propose the first fully representative IFA dataset – i.e., SPOTIFAI –, collected through extensive simulations over different scenarios in three network topologies. SPOTIFAI is publicly available for the research community. Then, we design GNN4IFA, a novel IFA detection technique that leverages GNNs. GNN4IFA is designed in two configurations – i.e., supervised (SAD) and unsupervised (UAD) – and is tested using SPOTIFAI. The results show that GNN4IFA can detect IFA in both configurations. In particular, SAD approaches 100% accuracy in all the setups, while UAD reaches a true positive rate of 100% and a negligible false positive rate. Additionally, comparing our GNN4IFA to ML-based state-of-the-art mechanisms, we confirmed an improvement of up to 6% in the detection accuracy for the supervised configuration and a 10% of true positive rate increase for the unsupervised configuration. Instead, compared to non-ML state-of-the-art mechanisms, GNN4IFA ensures a sharp improvement in the detection accuracy up to 20% and in the true positive rate up to 25%, for the supervised and unsupervised configuration, respectively. Lastly, we tested GNN4IFA generalisability by transferring models from one topology to the others. The results demonstrate UAD robustness and SAD limitations.

In the future we aim at extending GNN4IFA by designing a proactive IFA mitigation scheme relying on xAI techniques applied to GNN to identify routers mostly affected by the attack and blocking incoming traffic. Moreover, we plan to model GNN4IFA in a distributed fashion, designing an ad-hoc communication protocol, to validate the overhead introduced by our system. Lastly, we also plan to extend SPOTIFAI, by increasing setups heterogeneity and adding IFA leveraging non-existing content.

Acknowledgements

This paper was partially supported by *(i)* La Sapienza University of Rome within the Bando Ricerca 2020, project "Secure ANd PrivatE Information sharing (SANPEI)"; *(ii)* the CHIST-ERA IV project "EXPECTATION" (CHIST-ERA-19-XAI-005), co-funded by EU and the Italian MUR; *(iii)* project "SERICS" (PE00000014) under the NRRP MUR program funded by the EU - NGEU; and *(iv)* the European Commission under the Horizon Europe Programme, as part of the project "LAZARUS" (https://lazarus-he.eu/) (Grant Agreement no. 101070303).

We would also like to thank the anonymous reviewers for their comments and suggestions that helped us improve the quality of the paper.

References

[1] D. R. Cheriton and M. Gritter, “Triad: A New Next-Generation Internet Architecture,” 2000.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking Named Content,” in *International Conference on Emerging Networking Experiments and Technologies*, 2009, pp. 1–12.

[3] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, “Named Data Networking (NDN) Project,” *Relatório Técnico NDN-0001*, vol. 157, p. 158, 2010.

[4] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks,” *IEEE Communication Surveys and Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[5] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, “Interest Flooding Attack and Countermeasures in Named Data Networking,” in *IFIP Networking Conference*, 2013, pp. 1–9.

[6] A. Benarfa, M. Hassan, A. Compagno, E. Losiouk, M. B. Yagoubi, and M. Conti, “Chokifa: A New Detection and Mitigation Approach Against Interest Flooding Attacks in NDN,” in *International Conference on Wired/Wireless Internet Communication*, 2019, pp. 53–65.

[7] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, “Poseidon: Mitigating Interest Flooding DDoS Attacks in Named Data Networking,” in *IEEE Conference on Local Computer Networks*, 2013, pp. 630–638.

[8] S. Ramesh, C. Yaashuwanth, K. Prathibanandhi, A. R. Basha, and T. Jayasankar, “An optimized deep neural network based dos attack detection in wireless video sensor network,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2021.

[9] R. F. Fouladi, O. Ermiş, and E. Anarim, “A novel approach for distributed denial of service defense using continuous wavelet transform and convolutional neural network for software-defined network,” *Computers & Security*, vol. 112, p. 102524, 2022.

[10] A. Agiollo, M. Conti, P. Kaliyar, T.-N. Lin, and L. Pajola, “Detonar: Detection of routing attacks in rpl-based iot,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1178–1190, 2021.

[11] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, “Named Data Networking,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[12] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, “Naming in Content-Oriented Architectures,” in *ACM SIGCOMM Workshop on Information-Centric Networking, ICN*, 2011, pp. 1–6.

[13] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, “DoS and DDoS in Named Data Networking,” in *International Conference on Computer Communication and Networks, (ICCCN)*, 2013, pp. 1–7.

[14] V. Fung, J. Zhang, E. Juarez, and B. G. Sumpter, “Benchmarking Graph Neural Networks for Materials Chemistry,” *npj Computational Materials*, vol. 7, 2021.

[15] W. Fan, Y. Ma, Q. Li, Y. He, Y. E. Zhao, J. Tang, and D. Yin, “Graph Neural Networks for Social Recommendation,” in *The World Wide Web Conference, WWW*. ACM, 2019, pp. 417–426.

[16] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic Graph CNN for Learning on Point Clouds,” *ACM Transactions on Graphics*, vol. 38, pp. 146:1–146:12, 2019.

[17] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” in *Advances in Neural Information Processing Systems 3*, 2017, pp. 1024–1034.

[18] B. Yu, H. Yin, and Z. Zhu, “Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting,” in *International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 3634–3640.

[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A Comprehensive Survey on Graph Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4–24, 2021.

[20] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph Neural Networks: A Review of Methods and Applications,” *AI Open*, pp. 57–81, 2020.

[21] H. Dai, Y. Wang, J. Fan, and B. Liu, “Mitigate DDoS Attacks in NDN by Interest Traceback,” in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013, pp. 381–386.

[22] A. Benarfa, M. Hassan, E. Losiouk, A. Compagno, M. B. Yagoubi, and M. Conti, “Chokifa+: an early detection and mitigation approach against interest flooding attacks in ndn,” *International Journal of Information Security*, vol. 20, no. 3, pp. 269–285, 2021.

[23] Y. Xin, Y. Li, W. Wang, W. Li, and X. Chen, “A Novel Interest Flooding Attacks Detection and Countermeasure Scheme in NDN,” in *IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–7.

[24] H. Salah, J. Wulfheide, and T. Strufe, “Coordination supports security: A new defence mechanism against interest flooding in NDN,” in *40th IEEE Conference on Local Computer Networks, LCN 2015, Clearwater Beach, FL, USA, October 26-29, 2015*. IEEE Computer Society, 2015, pp. 73–81.

[25] H. Salah and T. Strufe, “Evaluating and Mitigating a Collusive Version of the Interest Flooding Attack in NDN,” in *IEEE Symposium on Computers and Communication (ISCC)*, 2016, pp. 938–945.

[26] K. Wang, H. Zhou, Y. Qin, and H. Zhang, “Cooperative-filter: countering interest flooding attacks in named data networking,” *Soft Computing*, vol. 18, no. 9, pp. 1803–1813, 2014.

[27] A. Benmoussa, A. E. K. Tahari, N. Lagraa, A. Lakas, F. Ahmad, R. Hussain, C. A. Kerrache, and F. Kurugollu, “A novel congestion-aware interest flooding attacks detection mechanism in named data networking,” in *28th International Conference on Computer Communication and Networks, ICCCN 2019, Valencia, Spain, July 29 - August 1, 2019*. IEEE, 2019, pp. 1–6.

[28] T. Zhi, Y. Liu, and Z. Yan, “An Entropy-SVM Based Interest Flooding Attack Detection Method in ICN,” in *IEEE Vehicular Technology Conference*, 2018, pp. 1–5.

[29] N. Kumar, A. K. Singh, and S. Srivastava, “Evaluating Machine Learning Algorithms for Detection of Interest Flooding Attack in Named Data Networking,” in *ACM International Conference Proceeding Series*, 2017, pp. 299–302.

[30] ——, “Feature Selection for Interest Flooding Attack in Named Data Networking,” *International Journal of Computers and Applications*, vol. 43, no. 6, pp. 537–546, 2021.

[31] J. Chen, G. Xing, M. Cui, H. Huo, and R. Hou, “Isolation forest based interest flooding attack detection mechanism in ndn,” in *2019 2nd International Conference on Hot Information-Centric Networking (HotICN)*. IEEE, 2019, pp. 58–62.

[32] I. J. King and H. H. Huang, “Euler: Detecting network lateral movement via scalable temporal graph link prediction,” in *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society, 2022.

[33] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, “THREATRACE: detecting and tracing host-based threats in node level through provenance graph learning,” *IEEE Transactions on Information and Forensics Security*, vol. 17, pp. 3972–3987, 2022.

[34] Y. Ji and H. H. Huang, “Nestedgnn: Detecting malicious network activity with nested graph neural networks,” in *IEEE International Conference on Communications, ICC 2022, Seoul, Korea, May 16-20, 2022*. IEEE, 2022, pp. 2694–2699.

[35] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, “On realistic network topologies for simulation,” in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, 2003, pp. 28–32.

[36] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP Topologies with Rocketfuel,” *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[37] F. Errica, M. Podda, D. Bacciu, and A. Micheli, “A fair comparison of graph neural networks for graph classification,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[38] W. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 257–266.

[39] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the Evolution of ndnSIM: An Open-Source Simulator for NDN Experimentation," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.

[40] L. A. Adamic and B. A. Huberman, "Zipf's Law and the Internet." *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[41] X. Zhang and R. Li, "A charging/rewarding mechanism-based interest flooding attack mitigation strategy in NDN," in *IFIP/IEEE International Symposium on Integrated Network Management, IM 2019, Washington, DC, USA, April 09-11, 2019*. IFIP, 2019, pp. 402–407.

[42] Z. Wu, W. Feng, M. Yue, X. Xu, and L. Liu, "Mitigation measures of collusive interest flooding attacks in named data networking," *Computers & Security*, vol. 97, p. 101971, 2020.

[43] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, "Strategies for Pre-training Graph Neural Networks," in *International Conference on Learning Representations, ICLR*, 2020.

[44] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "GPT-GNN: Generative Pre-Training of Graph Neural Networks," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2020, pp. 1857–1867.

[45] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017.

[46] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3837–3845.

[47] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology Adaptive Graph Convolutional Networks," *CoRR*, 2017.

[48] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" in *International Conference on Learning Representations, ICLR*, 2019.

[49] F. Wu, A. H. S. Jr, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying Graph Convolutional Networks," in *International Conference on Machine Learning, ICML*, vol. 97, 2019, pp. 6861–6871.

[50] Z. Wu, S. Peng, L. Liu, and M. Yue, "Detection of improved collusive interest flooding attacks using BO-GBM fusion algorithm in NDN," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 1, pp. 239–252, 2023.

[51] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *CoRR*, vol. abs/2012.15445, 2020.

[52] J. Ma, S. Ding, and Q. Mei, "Towards more practical adversarial attacks on graph neural networks," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[53] A. Varet and N. Larrieu, "How to generate realistic network traffic?" in *IEEE 38th Annual Computer Software and Applications Conference, COMPSAC 2014, Vasteras, Sweden, July 21-25, 2014*. IEEE Computer Society, 2014, pp. 299–304.

[54] J. Kepner, K. Cho, K. C. Claffy, V. Gadepally, P. Michaleas, and L. Milechin, "Hypersparse neural network analysis of large-scale internet traffic," in *2019 IEEE High Performance Extreme Computing Conference, HPEC 2019, Waltham, MA, USA, September 24-26, 2019*. IEEE, 2019, pp. 1–11.