Few Edges Are Enough: Few-Shot Network Attack Detection with Graph Neural Networks

IWSEC 2024, Kyoto, September 19th 2024

Tristan Bilot Nour El Madhoun Khaldoun Al Agha Anis Zouaoui

Université Paris-Saclay - LISN, Isep - LISITE, Iriguard

Motivating Example

Motivating Example

Normal activity within a network

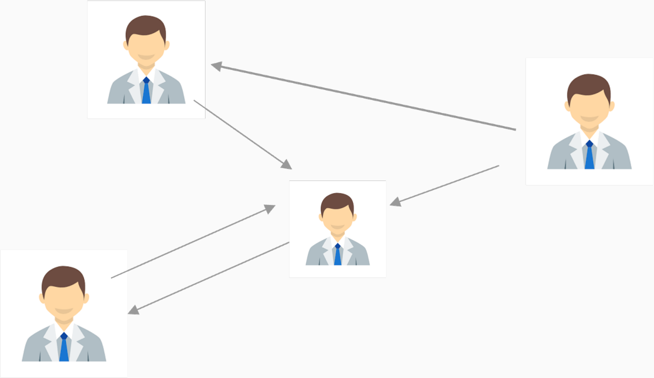


Figure 1: Example of normal communication between network hosts.

A malicious host attempts attack

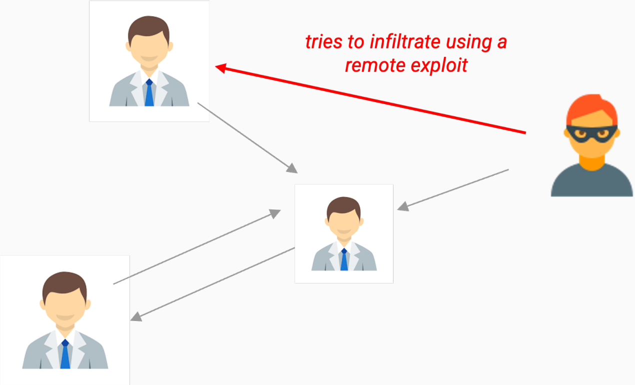


Figure 2: One host attempts attack on another host.

Motivating Example

Model the network as a graph

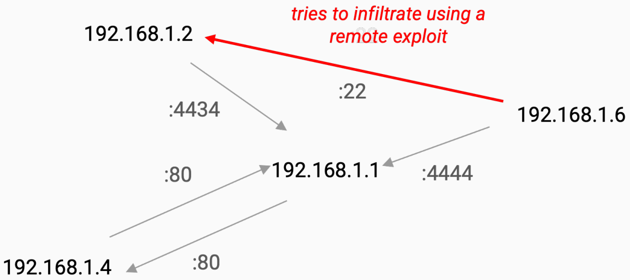


Figure 3: Such a network can be modelled as a graph, where nodes are IP addresses and edges are network flows. This example shows a user attempting to exploit another machine on a local network.

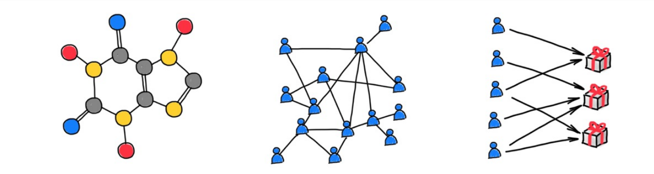
Motivating Example

Goal

- Detect various network attacks (e.g. DoS, scans, bruteforce, lateral movements, ...) leveraging the graph structure
- Reach high granularity (i.e. detection at the edge level)
- Reduce considerably the amount of hand-crafted labels
- Maintain a high precision with a low false positive rate

Current Methods

Most recent network-based attack detection methods use Graph Deep Learning and notably **Graph Neural Networks (GNNs)** due to their faculty to **capture complex and robust attack patterns** by leveraging the intrinsic graph structure of networks.



Current Methods

Current State-of-the-art (SOTA) methods can be classified in two main groups.

- Supervised Approaches
- Self-supervised Approaches

Challenges in Current Methods

Supervised Approaches (e.g. E-ResGAT, E-GraphSAGE)
Train the model to predict **labelled** edges/nodes from specific types of attacks.

Pros :

- Detect existing attacks with high precision

Cons :

- Require hand-crafted labels
- Do not generalize to new attacks, or variants of attacks

Self-supervised Approaches (e.g. Anomal-E)
Train the model to predict parts of the network activity and identify **clusters** of edges/nodes as outliers.

Pros :

- Do not labelled data for training the encoder

Cons :

- May not differentiate between benign anomalies and actual attacks

Current Methods

E-ResGAT [1]
Represents the graph as a line graph where each node is an edge, with features. Uses a Graph Attention Network (GAT) with residual connections to compute node embeddings. Trained in a **supervised** way.

E-GraphSAGE [2]
Aggregates information from neighbors using their edge features. Computes edge embeddings by concatenating node embeddings. Also trained in a **supervised** way.

Anomal-E [3]
Uses E-GraphSAGE as encoder and trains it in a **self-supervised** way by maximising/minimizing mutual information between training graphs and positively augmented and negatively augmented graphs, respectively.

Challenges in Current Methods

Limits of Anomal-E
We give particular attention to Anomal-E as it was at the time of writing this paper the only self-supervised approach to achieve SOTA results.

- Anomal-E uses self-supervised learning to train the GNN encoder with **both benign and attack data**
- The learned embeddings are decoded with an **Isolation Forest (IF)**, which is a one-class classifier **trained on benign edge only**.
- As a result, **it leads to a supervised method**, as benign and attack edges need to be identified
- We call these methods **benign-supervised**

Proposed Solution

Proposed Solution

Few-Shot Learning
We propose using **Few-Shot Learning (FSL)** as a balanced intermediary between fully supervised methods that cannot generalize to new attacks and fully unsupervised ones that yield too many false positives for a practical usage.

Pros :

- Requires only very few labelled examples
- Improved generalization compared to fully supervised methods

Cons :

- Still requires some historical attack data

Proposed Solution

Few Edges Are Enough (FEAE)

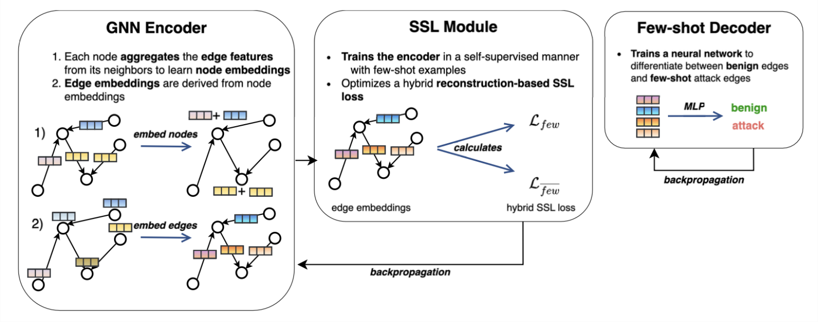


Figure 4: Architecture of FEAE

Proposed Solution

FEAE's Encoder

- We propose a **lightweight GNN encoder** layer that we can stack to capture spatial patterns from training network graphs
- We want to learn an **embedding vector for each edge**, which captures its semantic in the graph

Proposed Solution

FEAE's Encoder

- (1) Aggregate neighboring information via neighbors' edge features
- (2) Creates **node embeddings** from this aggregation
- (3) Derives **edge embeddings** by concatenating node embeddings

$$h_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} e_{uv}, \quad u \in N, \tag{1}$$

$$h_u = \sigma\left(h_{\mathcal{N}(u)} \mathbf{W}_{\text{agg}}\right), \tag{2}$$

$$h_{uv} = [h_u, h_v]\, \mathbf{W}_{\text{edge}}, \tag{3}$$

e_{uv} : feature vector of edge (u, v) ; $\mathcal{N}(u)$: neighboring nodes of node u ; $h_{\mathcal{N}(u)}$: sum aggregation u's neighboring edges ; h_u : embedding vector of node u ; σ : ReLU activation function ; $\mathbf{W}_{\text{agg}}$ and $\mathbf{W}_{\text{edge}}$: trainable weight matrices ; $[,]$: concatenation operation

Proposed Solution

FEAE's SSL Module

- At this point, we want to decode edge embeddings to **optimize a certain objective**
- We want to train the encoder to **differentiate between benign and malicious edges** using minimal labeled edges
- We propose a **hybrid SSL loss** that also integrates few-shot learning (FSL)

Proposed Solution

Contrastive-based loss

- **Goal** : create similar embeddings for edges with similar semantics
- (4-5) Calculates edge embeddings for the original graph G and an altered (negative) graph $\widetilde{G}$
- (6) Calculates a *summary* from G that summarizes all its semantics

$$\mathbf{H} = \mathrm{enc}(G), \tag{4}$$

$$\widetilde{\mathbf{H}} = \mathrm{enc}(\widetilde{G}), \tag{5}$$

$\mathbf{H}$ and $\widetilde{\mathbf{H}}$: edge embeddings of the graph and its negative augmentation, respectively ; $\widetilde{G} = \mathcal{A}(G)$ with $\mathcal{A}$ an augmentation function.

$$\vec{s} = \sigma\left(\mathcal{R}(\mathbf{H})\right), \tag{6}$$

$\mathcal{R}$ is the mean readout operation and σ is the sigmoid function.

Proposed Solution

Contrastive-based loss

- (7) Calculates the prob. of an edge to exist (positive)
- (8) Calculates the prob. of an edge to not exist (negative)
- (9) Computes Binary Cross Entropy (BCE) along all edges

$$\mathcal{D}\left(\mathbf{H}_{uv}, \vec{s}\right) = \sigma\left(\mathbf{H}_{uv}\mathbf{W}\vec{s}\right), \tag{7}$$

$$\mathcal{D}(\widetilde{\mathbf{H}}_{uv}, \vec{s}) = \sigma\left(\widetilde{\mathbf{H}}_{uv}\mathbf{W}\vec{s}\right), \tag{8}$$

$\mathcal{D}$: discriminator function, which returns the probability of an edge being either positive or negative ; $\mathbf{H}_{uv}$ and $\widetilde{\mathbf{H}}_{uv}$: respectively represent the positive and negative embeddings for (u, v).

$$\mathcal{L}_{\mathrm{c}} = -\mathbb{E}_{G}\left[\log\mathcal{D}\left(\mathbf{H}_{uv}, \vec{s}\right)\right] + \sum_{uv \in \widetilde{E}} \mathbb{E}_{\widetilde{G}}\left[\log\left(1 - \mathcal{D}\left(\widetilde{\mathbf{H}}_{uv}, \vec{s}\right)\right)\right], \tag{9}$$

E and $\widetilde{E}$ represent the edges in the positive graph and negative graph, respect

Proposed Solution

Reconstruction-based loss

- **Goal** : create different embeddings for the few-shot malicious edges
- (10) Reconstructs an approximation $\hat{\mathbf{X}}_{uv}$ of edge features $\mathbf{X}_{uv}$
- (11-12) Computes edge features' reconstruction error with MSE

$$\hat{\mathbf{X}}_{uv} = \sigma\left(\mathbf{H}_{uv}\mathbf{W}_{\text{rec}}\right), \tag{10}$$

$$\mathcal{L}_{\text{few}} = \sum_{uv \in \mathcal{E}_{\text{mal}}} \left(\mathbf{X}_{uv} - \hat{\mathbf{X}}_{uv}\right)^2, \tag{11}$$

$$\mathcal{L}_{\overline{\text{few}}} = \sum_{uv \in E \setminus \mathcal{E}_{\text{mal}}} \left(\mathbf{X}_{uv} - \hat{\mathbf{X}}_{uv}\right)^2, \tag{12}$$

$\mathcal{E}_{\text{mal}}$ is a set of k few-shot edges ; $E \setminus \mathcal{E}_{\text{mal}}$ is the set of remaining (unlabeled) non-few-shot edges, and $\mathbf{X}_{uv}$ represents the original features of edge (u, v) ; σ : sigmoid function ; $\mathbf{W}_{\text{rec}}$: weight matrix.

Proposed Solution

Reconstruction-based loss

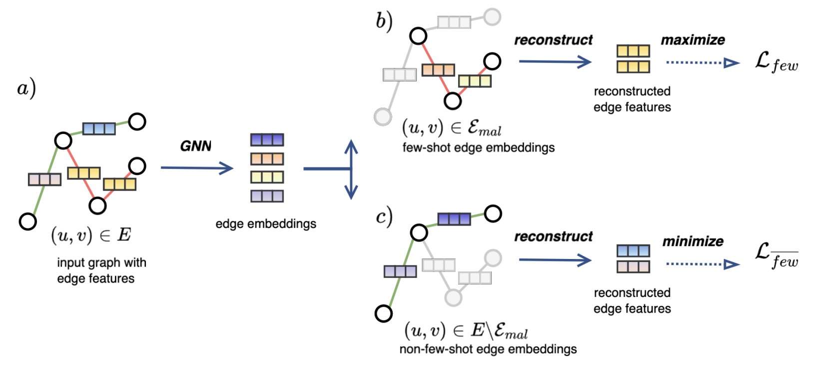


Figure 5: FEAE's reconstruction-based loss. a) calculation of edge embeddings by the GNN encoder ; b) maximization of the reconstruction error for few-shot edges ; c) minimization of the reconstruction error for all other edges.

Proposed Solution

Overall loss of the SSL module

$$\mathcal{L}_{\texttt{FEAE}} = \mathcal{L}_{\mathsf{c}} + \alpha \mathcal{L}_{\overline{\mathsf{few}}} - \beta \mathcal{L}_{\mathsf{few}} \tag{13}$$

where $\mathcal{L}_{\mathsf{c}}$ is the contrastive loss, $\mathcal{L}_{\overline{\mathsf{few}}}$ is the reconstruction loss of few-shot edges and $\mathcal{L}_{\mathsf{few}}$ is the loss of all other edges. α and β are trade-off coefficients to balance the reconstruction error of few-shot and non-few-shot examples.

We recommend to set $\alpha < \beta$, particularly when the dataset contains a significant number of malicious edges.

Proposed Solution

Decoder

- **Goal** : we aim to differentiate between malicious and benign edges given their embeddings
- (14) Simply feed the trained embeddings into a **MLP with sigmoid** activation to get a final prediction. The model is then trained to predict the few-shot edges as attack and all other edges as benign.
- Using this hybrid SSL loss allows to train the model effectively using **only very few labels**

$$\hat{\mathbf{y}} = \sigma\left(\mathrm{MLP}\left(\mathbf{H}\right)\right). \tag{14}$$

Evaluation

Evaluation

Datasets

- **NF-CSE-CIC-IDS2018-v2** [4]. This dataset is a Netflow version of the original CSE-CIC-IDS2018 dataset [5], containing approximately **18.9 million network flows**. Among these flows, around 12% correspond to attack samples, which are divided into **6 attack families** including *BruteForce, Bot, DoS, DDoS, Infiltration, Web attacks*.
- **NF-UNSW-NB15-v2** [4]. Also converted to Netflow format, this version of the UNSW-NB15 dataset [6] comprises **2.3 million flows**, with attack samples accounting for 4% of the dataset, distributed across **9 attack families** including *Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms*.

Evaluation

How the number of few-shot labels k impacts detection ?

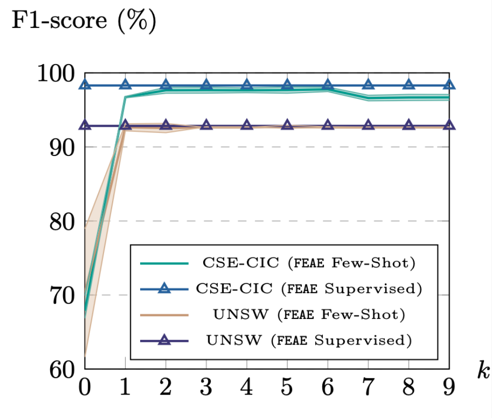


Figure 6: FEAE's performance with respect to k. Setting $k = 1$ is enough to approach the F1-score of fully supervised methods.

What looks the embedding space like ?

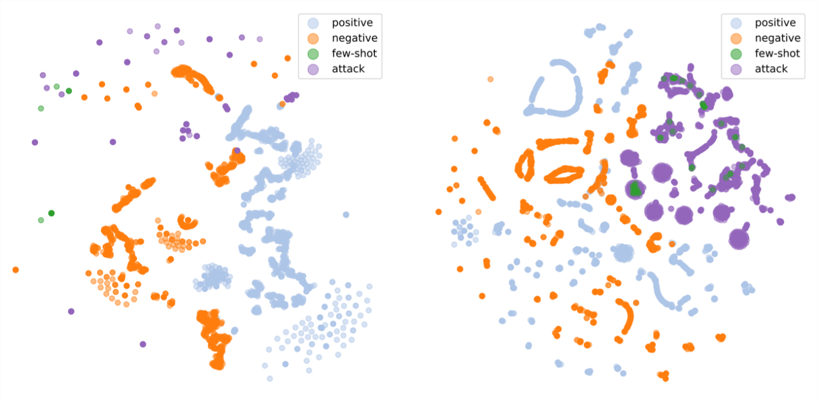


Figure 7: Left : Some edge embeddings produced by Anomal-E. Note that the few- shot edges are just for comparison as they are not leveraged in the original Anomal-E. **Right :** Edge embeddings generated by FEAE.

How FEAE performs compared to baselines ?

Data	Model	NF-CSE-CIC-IDS2018-v2			NF-UNSW-NB15-v2		
		F1	**Precision**	**Time**	**F1**	**Precision**	**Time**
A, X, Y	E-GraphSAGE	96.02	98.82	0.31	95.35	92.49	0.32
A, X, Y	LineGAT	93.84	96.84	4.3	95.33	91.81	14.2
A, X, Y	LineGCN	89.29	95.42	0.43	95.35	91.83	0.58
A, X, Y	LineSAGE	94.94	97.10	1.00	**95.90**	93.11	2.08
A, X, Y_{ben}	Anomal-E (IF)	94.46	96.86	85.1	$\boxed{91.14}$	$\boxed{85.78}$	9.2
A, X, Y_{ben}	Anomal-E (IF) + aug_1	$\boxed{96.53}$	$\boxed{98.84}$	81.3	87.38	84.13	7.9
A, X, Y_{few}	Anomal-E (Few-Shot)	95.3	97.28	24.5	92.47	86.42	1.45
A, X, Y_{few}	`FEAE`	96.40	99.12	19.6	92.60	89.56	1.22
A, X, Y_{few}	`FEAE` + aug_1	$\boxed{\textbf{97.44}}$	$\boxed{99.76}$	18.4	$\boxed{92.84}$	$\boxed{90.77}$	1.19

Figure 8: Experimental results. Colors : Supervised, Benign-supervised, Few-shot approaches.

Findings & Conclusion

Findings & Conclusion

Findings

- **Benign-supervised** approaches like Anomal-E yield **suboptimal results** and require **knowledge of all labels**
- Switching to a **few-shot** learning approach **improves detection precision**
- The architecture proposed in FEAE improves further performance using **only 1 edge label per attack family**

Future research

Future research work

- Evaluate the generalization capabilities of few-shot approaches to new attacks
- Improve scalability to very large networks

Questions

Thank you !

Do you have any questions ?

References i

L. Chang and P. Branco, "Graph-based solutions with residuals for intrusion detection : The modified e-graphsage and e-resgat algorithms," *arXiv preprint arXiv :2111.13597*, 2021.

W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-graphsage : A graph neural network based intrusion detection system for iot," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, IEEE, 2022.

E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-e : A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, 2022.

M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile networks and applications*, pp. 1–14, 2022.

I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

N. Moustafa and J. Slay, "Unsw-nb15 : a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, pp. 1–6, IEEE, 2015.