I. Comparison of the proposed Meta-IDS with Explainable AI-based E-GraphSAGE Model

E-GraphSAGE, a renowned explainable AI algorithm introduced by Weng et al. [1], utilize edge features to aggregate graph information through the input, aggregator function, and message passing mechanisms. We applied the E-GraphSAGE algorithm to the WUSTL-EHMS-2020, IoTID20, and WUSTL-IIOT-2021 datasets, utilizing only the provided edge features (EF) $e_{uv}, \forall_{uv} \in \mathcal{E}$. Since the datasets lack node features (NF), we initialized NF using the constant vector $X_v = 1, \ldots, 1$ as per Algorithm 1, maintaining the vector dimensions consistent with the number of EF.

$$h^k_{\mathcal{N}(\sqsubseteq)} = AGG_k(\{e^{k-1}_{u,v}, \forall u \in \mathcal{N}(v)\}) \tag{1}$$

The embedding of every node u in the vicinity of node v was combined to create embedding-node v at layer k, where $h^N_k(v)$ displays node embedding (u) in the preceding layer.

Eq. 9 shows the standard GraphSAGE model, but the E-GraphSAGE model used the aggregated embedding of the sample at adjacent edges of the k-th layer, as explained in the following equation.

$$h^k_{\mathcal{N}(\sqsubseteq)} = AGG_k(\{e^{k-1}_{u,v}, \forall u \in \mathcal{N}(v), uv \in \mathcal{E}\}) \tag{2}$$

where the edge-features uv from the $\mathcal{N}(v)$, sample adjacent-nodes v at layer $k-1$ are displayed by e^{k-1}_{uv}. Additionally, the edge-sample in the adjacent of $\mathcal{N}(v)$ is shown by $\{\forall u \in \mathcal{N}(v), uv \in \mathcal{E}\}$. the embedding-node v at k layer calculated by the original GraphSAGE algorithm is shown as follows.

$$h^k_v = \sigma(W^k.CONCAT(h^{k-1}_v, h^k_{\mathcal{N}(v)})) \tag{3}$$

The crucial distinction lies in the fact that the E-GraphSAGE algorithm computes $h^k_{\mathcal{N}(v)})$ using equation 11, which incorporates EF. Each graph node's k-hop neighbourhood provides the edge and topological information that is gathered and aggregated in the network-flow graph [1]. By concatenating the embedding of nodes u and v, it is possible to calculate the edge-embedding z^K_{uv} of edges uv, as illustrated below.

$$z^K_{uv} = CONCAT(z^K_u, z^K_v) uv \in \mathcal{E} \tag{4}$$

The E-GraphSAGE model comprises three primary steps: constructing a network graph using network-flow data, training a supervised model with the generated data, and creating edge embeddings to enable edge classification, discerning between normal and attack classes.

1) The construction of the network graph:: The network-flow data, comprising source and destination details alongside additional fields like data bytes and packet counts, naturally translates into a graph format. We define graph edges using four flow fields: Sport, Dport, SrcAddr, and DstAddr are used to identify nodes in the graph. DstAddr and Dport denote the destination node, while SrcAddr and Sport indicate the source node, exemplified by data exchange between 10.0.1.172:58059 and 10.0.1.150:1111. For network graph creation, we randomly selected source IP addresses ranging from 172.15.0.1 to 172.33.0.1 to obfuscate potential attack vectors. During graph construction, all other flow fields were assigned to the edge, resulting in featureless graph nodes. Each node received a vector containing all one values, as outlined in the algorithm.

2) E-GraphSAGE training:: Implemented a neural network comprising two layers of E-GraphSAGE (k=2), capturing data aggregated from a 2-hop neighborhood. The mean of edge features (EF) within the neighborhood sample is computed using the mean function, as defined below:

$$h^k_{\mathcal{N}(v)} = \sum_{u \in \mathcal{N}(v), uv \in \mathcal{E}} \frac{e^{k-1}_{uv}}{|\mathcal{N}(v)|_e} \tag{5}$$

Where $|\mathcal{N}(v)|_e$ shows the number of edges, e^{k-1}_{uv} shows edge features at layer $k-1$.

Utilized two E-GraphSAGE layers with 128 hidden units, ReLU activation, 20% dropout regularization, Adam optimizer, cross-entropy loss, and a learning rate of 0.001 for backpropagation. In the final layer, node embeddings were converted to edge embeddings by concatenating two node embeddings. These edge embeddings were then passed through a softmax layer during backpropagation to fine-tune trainable parameters by comparing them to dataset labels.

3) Edge classification:: After training and parameter tuning, the model categorizes unseen samples during evaluation. Test-flow records are transformed into graphs and passed through the trained E-GraphSAGE model. Edge embeddings are then generated, and class probabilities are computed using softmax and compared to actual class labels.

Algorithm 1 E-GraphSAGE edge embedding

Input: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$:
input edge features $\{e_{uv}, \forall_{uv}, \in \mathcal{E}\}$:
input node features $x_u = \{1, \ldots, 1\}$:
depth K :
weight matrices $W^k, \forall k \in \{1, \ldots, K\}$:
non-linearity σ :
differentiable aggregator function AGG_k :
Output: Edge embeddings $z_{u,v}, \forall_{uv}, \in \mathcal{E}$
$h^o_v \leftarrow x_v, \forall v \in \mathcal{V}$
for $k \leftarrow 1ToK$ **do**
  for $v \in \mathcal{V}$ **do**
    $h^k_{\mathcal{N}(\sqsubseteq)} \leftarrow AGG_k(\{e^{k-1}_{u,v}, \forall u \in \mathcal{N}(v), uv \in \mathcal{E}\})$
    $h^k_v \leftarrow \sigma(W^k.CONCAT(h^{k-1}_v, h^k_{\mathcal{N}(v)}))$
  end for
end for
$z_v = h^k_v$
for $uv \in \mathcal{E}$ **do**
  $z^K_{uv} \leftarrow CONCAT(z^K_u, z^K_v)$
end for

References

[1] Wai Weng Lo, Siamak Layeghy, Mohanad Sarhan, Marcus Gallagher, and Marius Portmann. E-graphsage: A graph neural network based intrusion detection system for iot. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2022.