1 Research Question

- **Research question**: Cross-platform binary code similarity detection.
- **Previous work**: Graph matching based algorithm. Can be slow, inaccurate, and hard to scale to other tasks.
- **Method**: Use deep graph models to generate embedding for the graph. In addition, use Siamese network to learn the difference.

2 Overview Figure

![Overview Figure](image)

3 Model

- Problem: Generate an embedding for the Attributed Control Flow Graph (ACFG).
• Structure2Vec model:

\[
\mu_v^{(t+1)} = F(x_v, \sum_{u \in N(v)} \mu_u^{(t)}), \forall v \in V \tag{1}
\]

\( F \) is a nonlinear transformation function.

• This paper:

\[
\mu_v^{(t+1)} = \tanh(W_1 x_v + \sigma(\sum_{u \in N(v)} \mu_u^{(t)})) \tag{2}
\]

Where \( \sigma \) is a \( n \) layer Fully connected neural network.

4 Siamese Learning

• Idea: learn how to generate embeddings by matching the generated embeddings of two similar inputs in training.

\[
\cos(\phi(g), \phi(g')) = \frac{\langle \phi(g), \phi(g') \rangle}{||\phi(g)|| \cdot ||\phi(g')||} \tag{3}
\]

5 Pre-Training:

• Learn on a general default task first: whether two binary functions are compiled from the same source code.

• Fine-tuning on specific tasks: Add additional data with specific task labels (i.e., A pair of graphs should be similar) into the data. Sample 50 times more often of the new data.
6 Evaluation

- Baselines:
  - Bipartite Graph Matching (BGM): Match the graphs directly using Bipartite Graph Matching algorithms.
  - Codebook-based Graph Embedding (Genius)[1]: Previous graph embedding method.

- Experiments:
  - Task 1: Compile OpenSSL on 3 different architectures, get 129,365 ACFGs.
  - Task 2: Large scale data with 33,045 firmware images.
  - Task 3: Vulnerable functions obtained from the vulnerability dataset
  - Task 4: Efficiency evaluation with different size of ACFGs.

- Result:
  - ROC curve:

![ROC curves for different approaches](image)

Figure 5: ROC curves for different approaches evaluated on the testing similarity dataset.

- ROC curve shows Gemini outperforms baselines.
- Efficiency: It’s faster than Genius[1].
Hyperparameters:

- Adding number of layers of MLP in the model doesn’t help much.
- When T=5 (Number of stacked embedding layers), the model gets best performance.

References

[1] Qian Feng, Rundong Zhou, Chengcheng Xu, Yao Cheng, Brian Testa, and Heng Yin. Scalable graph-based bug search for firmware images. In Pro-