The heavy reliance on third-party libraries in embedded firmware heightens software supply chain security risks. BCSD addresses known vulnerabilities, while reverse engineering reveals unknown ones.

**MMML for Binary Code**

- **Align** binary encoding with function names in latent space to generalize to zero-shot learning.
- **Reconstruct** high-level structures from binaries to assist in reverse engineering.
- **Generalize** to binaries across domains and different downstream tasks.

**Loss Functions**

\[
\mathcal{L}_{\text{Contrast}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(\alpha x_i^T y_i^T)}{\sum_{j=1}^{N} \exp(\alpha x_i^T y_j^T)} + \frac{1}{N} \sum_{j=1}^{N} \log \frac{\exp(\alpha y_j^T x_i^T)}{\sum_{i=1}^{N} \exp(\alpha y_j^T x_i^T)}
\]

\[
\mathcal{L}_{\text{Optim}} = -\frac{1}{N} \sum_{i=1}^{N} \log P_{\text{binary}}(y_i | x_i).
\]

\[
\mathcal{L}_{\text{Sim}} = \lambda \mathcal{L}_{\text{Contrast}} + \lambda \mathcal{L}_{\text{Optim}} - \mathcal{L}_{\text{Optim}}
\]

Here, \( x_i \) and \( y_i \) denote binary and function name embeddings in the \( i \)-th and \( j \)-th pairs. \( N \) represents the batch size, and \( \sigma \) is the temperature to scale the logits.

**BCSD Scenario**

Evaluate binary similarity by computing the cosine distance between embeddings generated by the trained binary encoder.

**Binary Distribution**

In small-scale function name generation experiments, we observed significant differences in results based on the splitting strategy:

- F1 score averaged 0.6640 when splitting by functions
- F1 score averaged 0.4708 when splitting by binaries

We also noticed poor generalization between binaries when evaluating other state-of-the-art approaches.

**References**