Practical Binary Code Similarity Detection with BERT-based Transferable Similarity Learning

Sunwoo Ahn
Department of Electrical and Computer Engineering
Seoul National University
Seoul, Korea

Seonggwan Ahn
Department of Electrical and Computer Engineering
Seoul National University
Seoul, Korea

Hyungjoon Koo
Department of Computer Science and Engineering
Sungkyunkwan University
Suwon, Korea

Yunheung Paek
Department of Electrical and Computer Engineering
Seoul National University
Seoul, Korea
Binary Code Similarity Detection (BCSD)

• BCSD Problem

• Many applications
  • Code clone detection
  • Malware detection
  • Malware family classification
  • Known vulnerability discovery
  • Code patching verification
Challenges

• Useful information is unavailable in a binary
  • e.g., variable name, structure, type, class hierarchy, etc.

• Binaries that have identical semantics can vary
  • compiler configuration, architecture, obfuscation, etc.

• Halting problem
  • Undecidable to prove
    the functional equivalency of two arbitrary programs
Existing Works

- Recent advances employ neural network with Siamese architecture

<table>
<thead>
<tr>
<th>Model</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gemini</td>
<td>GNN, Siamese NN</td>
</tr>
<tr>
<td>InnerEye</td>
<td>word2vec, LSTM, Siamese NN</td>
</tr>
<tr>
<td>Asm2Vec</td>
<td>PV-DM</td>
</tr>
<tr>
<td>PalmTree</td>
<td>BERT, GNN, Siamese NN</td>
</tr>
<tr>
<td>DeepSemantic</td>
<td>BERT, Softmax classifier</td>
</tr>
</tbody>
</table>
Existing Works

- Distance/loss function affects Siamese network (Marcelli et al., USENIX '22)

<table>
<thead>
<tr>
<th>Model</th>
<th>Distance function</th>
<th>Loss function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gemini</td>
<td>Cosine distance</td>
<td>Contrastive loss</td>
</tr>
<tr>
<td>InnerEye</td>
<td>Cosine distance</td>
<td>Contrastive loss</td>
</tr>
<tr>
<td>Asm2Vec</td>
<td>Cosine distance</td>
<td>Log probability</td>
</tr>
<tr>
<td>PalmTree</td>
<td>Cosine distance</td>
<td>Contrastive loss</td>
</tr>
<tr>
<td>DeepSemantic</td>
<td>None</td>
<td>Cross entropy</td>
</tr>
</tbody>
</table>

- Scalar value → oversimplification
Problem

• We question existing work in a realistic scenario
Our Main Approach

• Goal: improve performance for unseen dataset

• Transferable similarity learning (BERT-based)
  • Learning a relationship btw instructions with pre-training
  • Repeatedly showing good performance on an assembly language

• Better similarity detection: learning a weighted distance vector with a binary cross entropy
  • Weighted distance → relationships are represented in a vector
BinShot

① Preprocessing for Training Preparation
Pre-processor

- Executables (Corpus)
- Disassembled Functions
- Normalized Functions

② Building a Generic Model for Assembly

- Feed-forward Neural Network
- BERT
- [SOS], Ins 0, [MASK], Ins 2, [EOS]
- Logits
- Pre-trained BERT Model

③ Building a Special Model for Code Similarity

- Downstream Layers
- Similar
- Dissimilar
- Fine-tuned BERT Model

④ Detecting Similarity

- Function Embeddings
- Target Function
- Downstream Model
- Prediction: similar?
Experimental Setup

• Dataset
  • Compiled with 2 compiler (gcc, clang) & 4 optimization (O0-O3)
  • 1,400 binaries in total
    • GNU utilities – binutils, coreutils, diffutils, findutils
    • SPEC2006, SPEC2017
    • 11 Real-world programs (BusyBox, Libgmp, ...)

• Baseline models:
  • Gemini, Asm2vec, PalmTree, DeepSemantic
  • BinShot-CTR, BinShot
Evaluation - Effectiveness

- Evaluate whole dataset

- t-SNE visualization

![Graph showing evaluation metrics for different models](image-url)
Evaluation - Transferability

• Trained with SPEC 2006
Evaluation – Vulnerable Function Detection

- Realistic scenario setup
  - Database contains **vulnerable function embeddings**
  - Query binary is stripped
  - Goal: find a vulnerable function from a query binary

<table>
<thead>
<tr>
<th>Program</th>
<th>CVE</th>
<th>Vulnerable function</th>
<th>Gemini O0–O3 A/R</th>
<th>Asm2Vec O0–O3 A/R</th>
<th>PalmTree O0–O3 A/R</th>
<th>DeepSemantic O0–O3 A/R</th>
<th>BinShot-CTR O0–O3 A/R</th>
<th>BinShot O0–O3 A/R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2014-0221 [14]</td>
<td>dtls_process_heartbeat</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td></td>
<td>2015-1791 [17]</td>
<td>ssl3_get_new_session_ticket</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>NTP v4.2.7p10</td>
<td>2014-9295 [16]</td>
<td>crypto_recv</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctrl_putstr</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configure</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>Libav v0.8.3</td>
<td>2012-2776 [12]</td>
<td>decode_cell_data</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
</tbody>
</table>
Evaluation – Runtime Efficiency

• Runtime efficiency
  • Exp1 - Each function pair
  • Exp2 - 82300 function pairs (100 in database, 823 in query binary) with our predictor

<table>
<thead>
<tr>
<th>Model</th>
<th>Gemini (ms)</th>
<th>Asm2Vec</th>
<th>PalmTree</th>
<th>DeepSemantic</th>
<th>BinShot-CTR</th>
<th>BinShot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp1 (ms)</td>
<td>0.10</td>
<td>81.94</td>
<td>1.33</td>
<td>1.34</td>
<td>1.30</td>
<td>1.32</td>
</tr>
<tr>
<td>Exp2 (s)</td>
<td>1.16</td>
<td>6,734.66</td>
<td>29.03</td>
<td>1.51</td>
<td>1.45</td>
<td>1.54</td>
</tr>
</tbody>
</table>
Discussions & Limitations

• Mangled Names

• Function inlining

• Code obfuscation and other code constructs

• Rarely appeared instructions
Wrap-up

• Learning a weighted distance with a binary cross entropy improves robustness against unseen function pairs

• Superiority of BinShot
  • effectiveness, practicality (transferability & runtime)

• The other models but ours shows poor performance in a realistic scenario

• Open source project: https://github.com/asw0316/binshot
Thanks!