Probabilistic Naming of Functions in Stripped Binaries

James Patrick-Evans, Lorenzo Cavallaro, Johannes Kinder
Problem

- Reverse engineering is difficult
- Symbols relate binary objects to source code
- Useful for debugging software and reverse engineering
- Generic approach to labeling previously unseen functions
ELF Binaries

Analysis
- Function Boundaries
- Disassembly
- VEX IR
- Feature Extraction

Features

Probabilistic Fingerprint

Factor Graph

CRF

ELF Exporter

Evaluation
- Exact
- NLP
- Symbol2Vec

Probabilistic Naming of Functions in Stripped Binaries
Analysis

• Function Boundaries
  • Provided by ELF header (or externally via IDA Pro, Ghidra, Nucleus, or JIMA)
  • Extract every function in the .text section

• Static Analysis
  • Register classification, CFG, DFG
  • Function argument recovery
  • Transitive closure under callgraph

• Symbolic Analysis
  • Taint tracking
  • Data flows
  • Heap & stack analysis
Probabilistic Fingerprint

- Combine features in a vectorized form per function
  - Scalar types
  - Graphical types
  - Binary types
  - Dictionary types

- Incremental Principal Component Analysis

- Weighted Gaussian Naïve Bayes classifier

- Output the probability of each function having the name \( s \in S \)
Probabilistic Structural Inference

- Condition Random Field that models the conditional probability $p(y | x ; \theta)$.
- Operates over a factor graph with pairwise feature functions between observed known and unknown information.
- Feature functions are built from relationships extracted in our analysis.
Probabilistic Structural Inference II

• Feature functions ($f_p$) are discrete taking the value {0,1} and have a multiplicative weighting $\theta_p$.

• Weighting parameter $\theta$ is replicated globally per feature function.

• Approximate Inference is done via Loopy Belief Propagation passing messages along edges in the factor graph.

• Estimate the optimal parameters by maximizing the log likelihood of $\theta$ with Stochastic Gradient Descent.
Name Similarity

_\_LIBZ\_init\_conn_  \{initialize, connection\}  begin_networking_req.constp

1. regex, abbreviations
2. stemming, synsets

\{initi, connect\}  \{networking, begin, request\}

Similarity(x, y) = 0.79
Symbol2Vec

Evaluation

Probabilistic Naming of Functions in Stripped Binaries
Symbol2Vec II

Similarity($X, Y$) = $\frac{X \cdot Y}{\|X\| \|Y\|}$

Evaluation

opendir

readdir

dirfw

rewinddir

readdir_r

usb_bulk_send

usb_bulk_recv

send

recv

\[ \mathbf{a} \approx \mathbf{b} - \mathbf{c} + \mathbf{d} \]
Results

• Probabilistic Fingerprint
  • Small dataset of 149 programs from coreutils, moreutils, and x11-utils
  • Compiled under optimization levels og, o1, o2 using both clang and gcc resulted in over 2000 unique binaries
  • Comparison to IDA FLIRT, R2 Zignatures

• Probabilistic Structural Inference
  • 17,000 C ELF binaries from Debian Sid with debugging information
  • Comparison against Debin

<table>
<thead>
<tr>
<th>Tool</th>
<th>Exact</th>
<th>NLP</th>
<th>Symbol2Vec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debin</td>
<td>0.51</td>
<td>0.55</td>
<td>0.57</td>
</tr>
<tr>
<td>Punstrip</td>
<td>0.73</td>
<td>0.75</td>
<td>0.77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experiment</th>
<th>FLIRT</th>
<th>Zig</th>
<th>Punstrip</th>
</tr>
</thead>
<tbody>
<tr>
<td>clang, og → clang, og</td>
<td>0.29</td>
<td>0.14</td>
<td>0.94</td>
</tr>
<tr>
<td>clang, og → clang, o2</td>
<td>0.27</td>
<td>0.13</td>
<td>0.92</td>
</tr>
<tr>
<td>clang, og → gcc, o2</td>
<td>0.26</td>
<td>0.11</td>
<td>0.84</td>
</tr>
</tbody>
</table>
Summary

- Framework for binary analysis focusing on function naming in Python 3.
- High level probabilistic function fingerprint based on VEX IR.
- We build a probabilistic graphical model that reasons about all known information in a stripped binary to infer function names.
- We build and release Symbol2Vec, a function name representation built from all C binaries in Debian.

https://github.com/punstrip