Detecting Obfuscated Function Clones in Binaries using Machine Learning

Michael Pucher, Christian Kudera, Georg Merzdovnik
2022

University of Vienna / SBA Research
Security and Privacy / Applied Security
Function Clone Detection

- Binary function clones are common
- Vendoring/static linking of OSS
- Embedded firmware
- Malware also reuses code
- Use clone detection for version tracking

- But: Malware uses **obfuscation**
Contributions

- End-to-end function clone detection framework: **OFCI**
  - Rely on open-source tools
  - Make use of recent NLP research
- Method for preserving call-graph information: **Call-ID**
- Analysis of performance **issues with obfuscated clone detection**
- Analysis of **virtualized function clone detection** performance
Research Questions

• **RQ1**: Can existing approaches be reduced in complexity?
  - Large models in recent architectures
  - **Goal**: Maintain same level of performance

• **RQ2**: Can additional features improve detection results?
  - In this case: Preserve call graph information

• **RQ3**: Combined with dynamic analysis for virtualization obfuscation?
  - Has not been analyzed before
• Obfuscated Function Clone Identification (OFCI)
  • Full code largely not available for recent work
  • Provide framework for feature extraction from binaries
  • Implemented as Ghidra plugin
• Integrate ALBERT as language model
  • Recent approaches use BERT-based architectures for program analysis
  • OFCI uses ALBERT on disassembly text
  • Use PyTorch and Transformers for stock implementation
• Create instruction traces of virtualized code
  • Intel Pin
OFCI: Architecture Overview

Disassembler

Preprocess Data

Disassembler

Preprocess Data

Disassembler

Preprocess Data

Tracer

Embedding Inference

Pretraining

Finetuning

Inference

Pretrain Model

Finetune Model

model(f) = e

#
OFCl: Feature Extraction

LEA    RSI, [RBX + RBP*0x8]
MOV    EDI, R13D
XOR    R8D, R8D
SUB    EDI, R12D
MOV    ECX, 0x40f9e0
MOV    EDX, 0x40d0bc
CALL   <EXTERNAL>::getopt_long
CMP    EAX, -0x1
JZ     LAB_004019a8

LEA    RSI, [RBX + RBP * 08]  . MOV    EDI , R13D . XOR R8D , R8D . SUB
       EDI , R12D . MOV ECX , e0 f9 40 .
       MOV EDX , bc d0 40 . CALL 00 . CMP
       EAX , - 01 . JZ a8 19 40 .
OFCI: ALBERT

- BERT-like architectures are used in recent work
  - ALBERT reduces number of trainable parameters
- Pre-Training (Self-Supervised)
  - Train the network on large corpus of disassembly text
  - Masked language modeling: Mask tokens and have network guess them
- Fine-Tuning (Supervised)
  - Sample function pairs from the dataset
  - Assign labels (similarity)
  - Training objective: cosine similarity of embeddings = label
OFCl: Tracing

- Virtualization Obfuscation
  - Code is compiled into bytecode
  - Executed by small virtual machine
  - If functions differ in bytecode → clones not detectable statically

- **Idea:** Generate execution trace
  - Trace instructions
  - Implicitly capturing bytecode behavior

- Use Intel Pin and import in Ghidra
Evaluation

- Use dataset from recent work
  - Contains real-world O-LLVM obfuscated binaries
  - Create synthetic Tigress dataset for virtualization

- Train model
  - NVIDIA GTX Titan X for Pre-Training (1 week)
  - NVIDIA RTX 2070 Super for Fine-Tuning (24h)

- Compare against reported values
  - Similarity: ROC
  - Clone Search: Precision@1
Evaluation: Feature Extraction

- Ghidra Analysis is slow
  - Decompiler needed for full disassembly
  - Most expensive analysis
- However, parsing is fast
  - **SAFE** parsing performance is underreported
  - **OFCI** is still faster

<table>
<thead>
<tr>
<th></th>
<th>Time in Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binutils</td>
<td>TREX</td>
</tr>
<tr>
<td>PuTTY Function</td>
<td>SAFE</td>
</tr>
<tr>
<td>Findutils</td>
<td>OFCI</td>
</tr>
<tr>
<td>Diffutils</td>
<td></td>
</tr>
</tbody>
</table>

10^2

10^1

10
RQ1: Complexity Reduction

- Largest SotA model: **TREX**
  - 60M parameters, 700MB
- OFCI requires 17% of space at worst
  - When compared to similar models
    - 9M parameters, 35MB
- No speedup in evaluation use
RQ1: Obfuscations - ROC

Obfuscated Pairs

<table>
<thead>
<tr>
<th>Project</th>
<th>OBSCREX</th>
<th>OFICI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffutils</td>
<td>0.990</td>
<td>0.959</td>
</tr>
<tr>
<td>Findutils</td>
<td>0.990</td>
<td>0.967</td>
</tr>
<tr>
<td>GMP</td>
<td>0.990</td>
<td>0.869</td>
</tr>
<tr>
<td>ImageMagick</td>
<td>0.989</td>
<td>0.910</td>
</tr>
<tr>
<td>Libmicrohttpd</td>
<td>0.991</td>
<td>0.905</td>
</tr>
<tr>
<td>SQLite</td>
<td>0.993</td>
<td>0.956</td>
</tr>
<tr>
<td>Average</td>
<td>0.990</td>
<td>0.929</td>
</tr>
</tbody>
</table>
### RQ1: Obfuscations - Precision@1

<table>
<thead>
<tr>
<th>Obf. Approach</th>
<th>GMP</th>
<th>LibTomCrypt</th>
<th>ImageMagick</th>
<th>OpenSSL</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TREX</strong></td>
<td>0.926</td>
<td>0.938</td>
<td>0.934</td>
<td>0.898</td>
<td>0.924</td>
</tr>
<tr>
<td><strong>bcf</strong></td>
<td>0.802</td>
<td>0.920</td>
<td>0.933</td>
<td>0.883</td>
<td>0.885</td>
</tr>
<tr>
<td><strong>OFCl</strong></td>
<td>0.158</td>
<td>0.121</td>
<td>0.224</td>
<td>0.093</td>
<td>0.149</td>
</tr>
<tr>
<td><strong>TREX</strong></td>
<td>0.943</td>
<td>0.931</td>
<td>0.936</td>
<td>0.940</td>
<td>0.930</td>
</tr>
<tr>
<td><strong>cff</strong></td>
<td>0.772</td>
<td>0.920</td>
<td>0.890</td>
<td>0.795</td>
<td>0.844</td>
</tr>
<tr>
<td><strong>OFCl</strong></td>
<td>0.169</td>
<td>0.178</td>
<td>0.156</td>
<td>0.043</td>
<td>0.136</td>
</tr>
<tr>
<td><strong>TREX</strong></td>
<td>0.949</td>
<td>0.962</td>
<td>0.981</td>
<td>0.980</td>
<td>0.968</td>
</tr>
<tr>
<td><strong>sub</strong></td>
<td>0.940</td>
<td>0.960</td>
<td>0.981</td>
<td>0.961</td>
<td>0.961</td>
</tr>
<tr>
<td><strong>OFCl</strong></td>
<td>0.249</td>
<td>0.214</td>
<td>0.283</td>
<td>0.169</td>
<td>0.229</td>
</tr>
</tbody>
</table>
RQ1: Function Search Performance

- Reduction of model complexity
  - Other approaches use additional features (e.g. TREX and microtraces)
- Definition of similarity
  - Datasets rely on function names
  - Same name and different semantics?
- Dataset differences
  - Dataset is from TREX, but TREX doesn’t use the whole dataset
Function Fragments

- All approaches have input limits
  - Discard everything after limit
- OFCI combines multiple fragments
  - Steep drop if > 1 fragment
- Different approaches needed
  - Transformers too expensive
RQ2: Call-ID

- Does Call-ID have an effect?
  - Yes, but slight
  - Slightly favorable across all ROC-AUC measurements
- Effect is bigger with more calls
- Bigger effect needs model redesign
RQ2: Call-ID - Precision@1

![Chart showing Precision@1 with and without Call-ID for Diffutils, Findutils, GMP, ImageMagick, Libmicrohttpd, and SQLite.](chart.png)
RQ3: Virtualization Obfuscated Code

- ROC-AUC close to random
  - Precision@1: 0 for O0-Virt
  - Precision slightly above random for Cross-Virt
- Random results here can affect results of other evaluations
RQ3: Virtualized Code - Issues

- Issues with dataset
  - Synthetic dataset
  - No real-world functions
- Synthetic functions are too "simple"
  - No changes between O2 and O3
- Only small differences in traces
  - Also: Input size limitations!
Conclusion

- Introduced the **OFCI** framework
  - Efficient feature extraction
  - Open-source tools
  - Reduced model size
- Introduced **Call-ID**
  - Slight effect, worth pursuing in future work
- Analysis of performance **issues with obfuscated clone detection**
  - Issues with training data selection and similarity definition
- Analysis of **virtualized function clone detection** performance
  - Tracing approach not effective without modification
Questions?
Backup Slides
RQ3: Virtualized Code - Dataset Generation

- New approach for dataset generation is needed
  - Large-scale (real-world) datasets with Tigress currently not possible
  - Not possible to apply to popular open-source binaries
  - Needs re-design of Tigress

- Models working with less data instead?
  - There will always be cases like Tigress...
<table>
<thead>
<tr>
<th>Category</th>
<th>Analysis</th>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>O0</td>
<td>88</td>
<td>8</td>
</tr>
<tr>
<td>O1</td>
<td>95</td>
<td>6</td>
</tr>
<tr>
<td>O2</td>
<td>90</td>
<td>6</td>
</tr>
<tr>
<td>O3</td>
<td>103</td>
<td>7</td>
</tr>
<tr>
<td>BCF</td>
<td>102</td>
<td>6</td>
</tr>
<tr>
<td>CFF</td>
<td>36</td>
<td>7</td>
</tr>
<tr>
<td>IBR</td>
<td>410</td>
<td>5</td>
</tr>
<tr>
<td>SPL</td>
<td>124</td>
<td>7</td>
</tr>
<tr>
<td>SUB</td>
<td>57</td>
<td>6</td>
</tr>
<tr>
<td>EA</td>
<td>42</td>
<td>11</td>
</tr>
<tr>
<td>VIRT</td>
<td>248</td>
<td>17</td>
</tr>
<tr>
<td>VIRT-EA</td>
<td>271</td>
<td>17</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>hidden_size</td>
<td>768</td>
<td></td>
</tr>
<tr>
<td>intermediate_size</td>
<td>3072</td>
<td></td>
</tr>
<tr>
<td>num_attention_heads</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>max_position_embeddings</td>
<td>514</td>
<td></td>
</tr>
<tr>
<td>num_hidden_layers</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>vocab_size</td>
<td>868</td>
<td></td>
</tr>
<tr>
<td>pretrain_batch_size</td>
<td>520</td>
<td></td>
</tr>
<tr>
<td>finetune_batch_size</td>
<td>522</td>
<td></td>
</tr>
<tr>
<td>peak_learning_rate</td>
<td>0.00005</td>
<td></td>
</tr>
<tr>
<td>transformers_version</td>
<td>4.11.3</td>
<td></td>
</tr>
</tbody>
</table>
Transformers

INPUT: je suis étudiant

OUTPUT: I am a student
Transformers: Self-Attention