Black-box Code Analysis for Reverse Engineering

Grégoire Menguy, CEA LIST, France
Sébastien Bardin, CEA LIST, France
Nadjib Lazaar, LIRMM, France
Arnaud Gotlieb, Simula, Norway
Richard Bonichon, Tweag IO, France
Cauim de Souza Lima, CEA LIST, France
Grégoire Menguy
PhD student at CEA LIST  @BinsecTool

@grmenguy

https://gregoiremenguy.github.io/
Context

Software size $\uparrow$ +

- Hard to verify code
- Hard to test code
- Hard to understand code

Need for automatic software analysis
White vs Black-box

White-box analysis

- Rely on the syntax
  - Symbolic exec.
  - Abstract interp.
- Scale issues

Black-box analysis

- Rely on code executions
- Insensitive to syntactic complexity

Séminaire IMT 2022
Scenarios

Deobfuscation
- Malware analysis
- Assess obfuscation strength

Contract inference
- Core refactoring
- Code understanding
Search-based Local Blackbox Deobfuscation: Understand, Improve and Mitigate

Grégoire Menguy, CEA LIST, France
Sébastien Bardin, CEA LIST, France
Richard Bonichon, Tweag IO, France
Cauim de Souza Lima, CEA LIST, France
Obfuscation

\[ P_1 \]
int f(in * l);
int main();

\[ P_2 \]
double L,o,P,
=dt,T,Z,D=1,d,
s[999],E,h= 8,
I,J,K,w[999],M,
m,0,n[999],j=  

Deobfuscation
Deobfuscation

Protecting Software through Obfuscation: Can It Keep Pace with Progress in Code Analysis?

SEBASTIAN SCHRITTWIESER, St. Pölten University of Applied Sciences, Austria
STEFAN KATZENBEISSER, Technische Universität Darmstadt, Germany
JOHANNES KINDER, Royal Holloway, University of London, United Kingdom
GEORG MERZDOVNIK and EDGAR WEIPPL, SBA Research, Vienna, Austria

A Generic Approach to Automatic Deobfuscation of Executable Code

Babak Yadegari  Brian Johannesmeyer  Benjamin Whitely  Saumya Debray
Department of Computer Science
The University of Arizona
Tucson, AZ 85721
{babaky, bjohannesmeyer, whitely, debray}@cs.arizona.edu

Symbolic deobfuscation: from virtualized code back to the original*

Jonathan Salwan¹, Sébastien Bardin², and Marie-Laure Potet³

Backward-Bounded DSE: Targeting Infeasibility Questions on Obfuscated Codes*

Sébastien Bardin
CEA, LIST, 91191 Gif-Sur-Yvette, France
sebastien.bardin@cea.fr

Robin David
CEA, LIST, 91191 Gif-Sur-Yvette, France
robin.david@cea.fr

Jean-Yves Marion
Université de Lorraine, CNRS and Inria, LORIA, France
jean-yves.marion@loria.fr

BINSEC

TRI L O N
Dynamic Binary Analysis

MIA SM
Deobfuscation

White-box deobfuscation is highly efficient
Anti-White-Box Deobfuscation

But efficient countermeasures

Information Hiding in Software with Mixed Boolean-Arithmetic Transforms

Yongxin Zhou, Alec Main, Yuan X. Gu, and Harold Johnson
Cloakware Inc., USA
{yongxin.zhou, alec.main, yuan.gu, harold.johnson}@cloakware.com

How to Kill Symbolic Deobfuscation for Free (or: Unleashing the Potential of Path-Oriented Protections)

Mathilde Ollivier
CEA, LIST, Paris-Saclay, France
mathilde.ollivier2@cea.fr

Richard Bonichon
CEA, LIST, Paris-Saclay, France
richard.bonichon@cea.fr

Sébastien Bardin
CEA, LIST, Paris-Saclay, France
sebastien.bardin@cea.fr

Jean-Yves Marion
Université de Lorraine, CNRS, LORIA
Nancy, France
Jean-Yves.Marion@loria.fr

Probabilistic Obfuscation through Covert Channels

Jon Stephens
Babak Yadegari
Christian Collberg
Saumya Debray
Carlos Scheidegger
Department of Computer Science
The University of Arizona
Tucson, AZ 85721, USA
Email: {stephensj2, babaky, collberg, debray, cscheiden}@cs.arizona.edu
New Threat: Black-box Deobfuscation

Bypasses white-box methods limitations
### Open Questions

<table>
<thead>
<tr>
<th>Understand</th>
<th>Improve</th>
<th>Mitigate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strenghts ?</td>
<td>Why MCTS ?</td>
<td>How to protect ?</td>
</tr>
<tr>
<td>Weaknesses ?</td>
<td>Can be improved ?</td>
<td></td>
</tr>
<tr>
<td>Why ?</td>
<td>Impacted by SOTA protections ?</td>
<td></td>
</tr>
</tbody>
</table>
Contributions

Understand

- Propose missing formalisation
- Refine Syntia Xps: new strengths & weaknesses
- Show & explain why MCTS not appropriate

Improve

- S-metaheuristics > MCTS
- Implement our approach: Xyntia
- Evaluation of Xyntia

Mitigate

- Propose 2 protections
- Evaluate them against Xyntia and Syntia
Black-box deobfuscation
What’s that?
Black-box Deobfuscation

1. Sample

- $(t = 1, T = 2) \rightarrow -1$
- $(t = 2, T = 5) \rightarrow -3$
- $(t = 0, T = 6) \rightarrow -6$

2. Learn

- $(t = 1, T = 2) \rightarrow -1$
- $(t = 2, T = 5) \rightarrow -3$
- $(t = 0, T = 6) \rightarrow -6$

$t - T$
Learning Engine

Expression grammar

\[ U := U + U | U - U | U \times U | t | T | 1 \]

\[
U \times U \quad t + T
\]

\[
t - U
\]

\[
expr \quad \rightarrow \quad \Delta
\[
\begin{align*}
expr(t = 1, T = 2), & -1 \\
expr(t = 2, T = 5), & -3 \\
expr(t = 0, T = 6), & -6 \\
\ldots
\end{align*}
\]
Why Black-box?

Given a language $L$ and an expression “$e$” in $L$

<table>
<thead>
<tr>
<th>Syntactic complexity</th>
<th>Semantic complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of the expression “$e$”</td>
<td>Size of the smallest expr. in $L$ equivalent to “$e$”</td>
</tr>
</tbody>
</table>

Example

$t - T$ is syntactically simpler than $(t \lor -2T) \times 2 - (t \oplus -2T) + T$

**but** they share the same semantic complexity (being equivalent)
Why Black-box?

Given a language $L$ and an expression “$e$” in $L$

<table>
<thead>
<tr>
<th>Syntactic complexity</th>
<th>Semantic complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of the expression “$e$”</td>
<td>Size of the smallest expr. in $L$ equivalent to “$e$”</td>
</tr>
</tbody>
</table>

Example

$t - T$ is syntactically simpler than $(t \lor -2T) \times 2 - (t \oplus -2T) + T$

**but** they share the same semantic complexity (being equivalent)

Obfuscation increases syntactic complexity

→ No impact on black-box methods
Understand
Zoom of SOTA: Syntia

Dig into Syntia and deepen its evaluation

- RQ1: stability of Syntia
- RQ2: efficiency of Syntia
- RQ3: impact of operators set
Syntia: New Results

Stable

Quality

Correctness

Speed

Robustness
Experimental Design

B1 (Syntia)
- 500 expressions
- Up to 3 inputs
- Redundancy
- Unbalances w.r.t. type

B2 (ours)
- 1110 expressions
- 2 to 6 inputs
- No redundancy
- Balances w.r.t. type

<table>
<thead>
<tr>
<th>Type</th>
<th># Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Bool.</td>
<td>150</td>
</tr>
<tr>
<td>Arith.</td>
<td></td>
</tr>
<tr>
<td>MBA</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Distribution of samples in benchmark B2
Evaluation of Syntia

B1 (Syntia)

- With a 60 s/expr. timeout: 75% of success rate
- With a 1 h/expr. timeout: 88% of success rate
- With a 12 h/expr. timeout: 97% of success rate

B2 (Ours)

<table>
<thead>
<tr>
<th>Table 2: Syntia depending on the timeout per expression (B2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Succ. Rate</td>
</tr>
<tr>
<td>Equiv. Range</td>
</tr>
<tr>
<td>Mean Qual</td>
</tr>
</tbody>
</table>
Why?

- Syntia manipulates non terminal expressions, e.g., $U - V$
- Scoring of non terminal expressions can be misleading
  
  \[
  U - V \sim \begin{cases} 
  t - T \\
  t - 1 \\
  1 - 1
  \end{cases}
  \]

- Syntia (i.e., MCTS) = “almost BFS”
Improve
Syntia sees blackbox deobfuscation as a single player game.

We propose to see it as an optimization problem.

Goal: find $s^*$ s.t. $f(s^*) \leq f(s)$, $\forall s \in S$
S-metaheuristics

- Solve optimization problems
S-metaheruristics

– Solve optimization problems
S-metaheuristics

- Solve optimization problems
S-meteheuristics

- Solve optimization problems
S-metaheuristics

- Solve optimization problems

Only terminal expressions
New Prototype: Xyntia

S-metaheuristics

Can choose between:
- Hill Climbing
- Simulated Annealing
- Metropolis Hasting
- Iterated Local Search

\[ \text{MCTS} \]
Xyntia vs Syntia

B1 (Syntia)

– 100 % success rate in 1s/expr.

B2 (Ours)

Syntia: 75% in 60 s/expr.
Xyntia vs Syntia

- 100% success rate in 1s/expr.

Syntia: 75% in 60 s/expr.

Robust & Fast

Stable
Correct
Good quality
Other Experiments

- Xyntia vs Qsynth
- Xyntia vs “compiler like simplifications”
- Xyntia vs program synthesizer CVC4
- Xyntia vs superoptimizer STOKE

- Use-cases
  - State-of-the-art protections
  - VM-based obfuscation
What’s Next?

PROTECTIONS ARE BROKEN!

DON'T WORRY...

I HAVE A PLAN
Mitigate
Context: Virtualization

Fig. 1: Virtualization based obfuscation

Proved to be sensitive to black-box deobfuscation
Why VM-based Obf. Is Vulnerable?

- Handlers are too semantically simple
  \[ +, -, \times, \wedge, \lor \]
- Obfuscation increases syntactic complexity
  \[ \Rightarrow \]
  Black-box deobf. is not impacted

We need to move ...

From syntactic to semantic complexity
Semantically Complex Expressions

Goal

- Increase the semantic complexity of each handler
- Keep a Turing complete set of handlers

Example

\[
\begin{align*}
    h_0 &= (x + y) + \neg((a - x^2) - (xy)) \\
    +\ h_1 &= (a - x^2) - xy + (- (y - (a \wedge x)) \times (y \otimes x)) \\
    +\ h_2 &= (y - (a \wedge x)) \times (y \otimes x) \\
    h &= x + y
\end{align*}
\]
Merged Handlers

Goal: Increase handlers semantic complexity + sampling harder

Example: \( h_1(x, y) = x + y \) and \( h_2(x, y) = x \land y \)

\[ h(x, y, c) = \text{if } (c = \text{cst}) \text{ then } h_1(x, y) \text{ else } h_2(x, y) \]

Hide conditionals:

```c
int32_t h(int32_t a, int32_t b, int32_t c) {
    // if (c == \text{cst}) then \( h_1(a, b, c) \) else \( h_2(a, b, c) \);
    int32_t res = c - \text{cst} ;
    int32_t s = res >> 31;
    res = ((res ^ s) -s) >> 31) & 1;
    return h1(a, b, c)*(1 - res) + res*h2(a, b, c);
}
```
Semantically Complex Handlers: Results

More results: Syntia with 12h/exprs. $\rightarrow$ 1/15 on BP1
Merged Handlers: Results

Figure 10: Merged handlers: Xyntia (timeout=60s)

More results: Syntia finds nothing for $\geq 2$ nested ITE
Xyntia: The Recap

MCTS is not appropriate for blackbox deobfuscation
→ Search space too unstable
→ Estimation of non terminal expressions pertinence is misleading

S-metaheuristics yields a significant improvement
→ More robust
→ Much Faster

Moving for syntactic to semantic complexity
→ 2 efficient methods to protect against blackbox deobfuscation
Discussion: No Guarantees

– Xyntia & Syntia have no correctness guarantees

– But some contexts need it:
  - Code verification
  - Code refactoring
  
Can black-box approaches have clear guarantees?
Automated Program Analysis: Revisiting Precondition Inference through Constraint Acquisition

Grégoire Menguy, CEA LIST, France
Sébastien Bardin, CEA LIST, France
Nadjib Lazaar, LIRMM, France
Arnaud Gotlieb, Simula, Norway
Dream: Infer Preconditions

Inputs

\[
\text{find first of}(\text{int* a, int m, int* b, int n})
\]

Description: returns the index of the first element in “a” (of size “m”) present in “b” (of size “n”)

Outputs

Postcond.

\[
\text{ERROR!}
\]
Dream: Infer Preconditions

\[ m > 0 \Rightarrow valid(a) \] \\
\& \\
\[ m > 0 \land n > 0 \Rightarrow valid(b) \]

**Inputs**
- \( m > 0 \)
- \( n > 0 \)
- \( m \) is the size of a
- \( n \) is the size of b

**Outputs**
- \( find\_first\_of(int* a, int m, int* b, int n) \)

Description: returns the index of the first element in “a” (of size “m”) present in “b” (of size “n”)

Undecidable problem: Rice theorem (1953)
Active Constraint Acquisition

Background knowledge: rules to speed up learning

\[ B : \{ \text{constr} \} \]

\[ K \]

true

\[ B \]
Active Constraint Acquisition

Query
Elise: 8h – 12h
Paul: 10h – 11h
...

yes / no

$B : \{ \text{constr} \} \cup K$

true

no: Top-down

yes: Bottom-up

Séminaire IMT 2022
Active Constraint Acquisition
Careful: too many queries
Theoretical Analysis

PreCA guarantees

- If B is expressive enough
- If code exec. always terminates

These are good theoretical guarantees

SOTA executions based methods, from programming language community, have no clear guarantees
Evaluation

**Dataset:** 94 learning tasks • compiled C functions (string.h, arrays, arithmetic ...)

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>PreCA</th>
<th>Daikon, PIE, Gehr et al</th>
<th>P-Gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q = true$</td>
<td>92%</td>
<td>At most 52%</td>
<td>74%</td>
</tr>
<tr>
<td>$Q \neq true$</td>
<td>41%</td>
<td>At most 23%</td>
<td>34%</td>
</tr>
</tbody>
</table>

PreCA better in 5s than concurrent tools in 1 hour
Conclusion

Black-box methods can be used in broad contexts

Deobfuscation – Contracts inference

Balance between robustness and guarantees
Thank you for your attention

@grmenguy

https://gregoiremenguy.github.io/