

Industrial Approach: Obfuscating Transformations

Yury Lifshits

Steklov Institute of Mathematics, St.Petersburg, Russia
yura@logic.pdmi.ras.ru

Tartu University
17/03/2006

Commercial Obfuscators:

- Semantic Designs: Thicket™ obfuscators
<http://www.semanticdesigns.com/Products/Obfuscators/>
- Zelix Klassmaster™ obfuscator
<http://www.zelix.com/klassmaster/>
- PreEmptive: DotObfuscator™
<http://www.preemptive.com/products/dotfuscator/>
- Only for Java: at least 26 obfuscators
http://dmoz.org/Computers/Programming/Languages/Java/Development_Tools/Obfuscators/

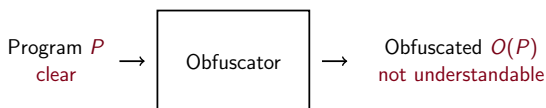
Yury Lifshits (Steklov Inst. of Math) Obfuscating Transformations Tartu '06 1 / 26

Outline

- 1 How to Develop an Obfuscator?
 - Anatomy of Obfuscator
 - Quality of Obfuscator
- 2 Library of Obfuscating Transformations
 - Data Obfuscation
 - Control Flow Obfuscation
 - Advanced Techniques

Yury Lifshits (Steklov Inst. of Math) Obfuscating Transformations Tartu '06 3 / 26

Objectives of Obfuscator



Objectives:

- Make automated analysis difficult
- Make code more complicated
- Make decompilation & reverse engineering difficult
- Make code not readable by human

Yury Lifshits (Steklov Inst. of Math) Obfuscating Transformations Tartu '06 5 / 26

Anatomy of Obfuscator (2)

The workflow of obfuscator:

- Parse input program
 - Makes a list of obfuscation candidates: classes, variables, methods
 - Constructs internal representation of the program (e.g. control flow and basic blocks)
 - Makes some **appropriateness** suggestions
- Main while loop (until constraints are exceeded or quality is achieved)
 - Choose next (by priority) element of the program to be obfuscated
 - Implement appropriate obfuscating transformation (from obfuscator library)
 - Update internal representation

Yury Lifshits (Steklov Inst. of Math) Obfuscating Transformations Tartu '06 7 / 26

Outline

- 1 How to Develop an Obfuscator?
 - Anatomy of Obfuscator
 - Quality of Obfuscator
- 2 Library of Obfuscating Transformations
 - Data Obfuscation
 - Control Flow Obfuscation
 - Advanced Techniques

Yury Lifshits (Steklov Inst. of Math) Obfuscating Transformations Tartu '06 4 / 26

Anatomy of Obfuscator (1)

How real obfuscator works?

- 1 Prepares program to be obfuscated
- 2 Makes a single transformation
- 3 Repeats **step 2** until task completed or constraints exceeded

Yury Lifshits (Steklov Inst. of Math) Obfuscating Transformations Tartu '06 6 / 26

Quality of Obfuscation

How good is obfuscation? Measures:

- Potency
 $\frac{\text{Complexity}(O(P))}{\text{Complexity}(P)}$
- Resilience (irreversibility)
Weak, strong, one-way
- Cost
Slowdown, increasing of code size and space requirements
- Stealth
How similar are introduced obfuscated constructions to the rest of the code

Yury Lifshits (Steklov Inst. of Math) Obfuscating Transformations Tartu '06 8 / 26

Software Complexity Metrics

How do you define a program code complexity?

- Program length
 - Number of operators and operands
- Data flow complexity
 - Number of inter-block variable references
- Cyclomatic complexity
 - Number of predicates in a function
- Nesting complexity
 - Number of nesting level of conditionals in a program
- Data structure complexity
 - Complexity of the static data structures in the program like variables, vectors, records
- OO Metrics
 - Level of inheritance, coupling, number of methods triggered by another method, non-cohesiveness

Statistical Metrics

Measuring chaos:

- Distribution of opcodes (and any elements of program)
 - Rare elements contain information. Replace them by basic instructions
- Clustering (usage of variables, control flow commands)
 - Best of all: no clustering, uniform distribution
- Code patterns
 - Destroy long repeating patterns in program

Cost Analysis

What do we pay for security?

- Costs at creation time
 - Obfuscation need time!
- Costs at transmission time (resulting size)
 - Inlining library functions may increase size enormously!
- Cost at execution time
 - Checking procedures, dummy code, inlining
- Cost by not using efficiency enhancing mechanisms
 - Caching is rarely possible; losing module structure

Outline

- 1 How to Develop an Obfuscator?
 - Anatomy of Obfuscator
 - Quality of Obfuscator
- 2 Library of Obfuscating Transformations
 - Data Obfuscation
 - Control Flow Obfuscation
 - Advanced Techniques

Top Three Methods

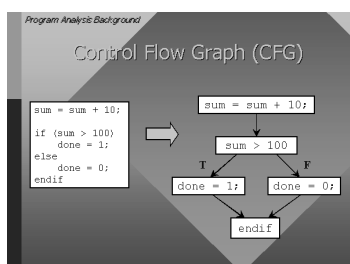
- Renaming
 - variables/procedures/classes/methods
- Deleting comments and spaces
 - (destroying layout)
- Inserting dead code

Data Obfuscation

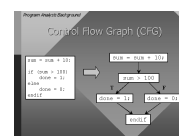
Any ideas for data obfuscation?

- Variable splitting
- Scalar/object conversion
- Static data to procedure
- Change variable lifetime
- Split/fold/merge arrays
- Change encoding
- Merge scalar variables

Control Flow (1)



Control Flow (2)



Compiler theory: program = control flow graph (CFG)

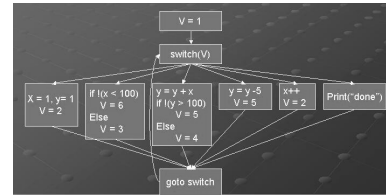
- Node = basic block = straight-line piece of code without any jumps or jump targets
- Directed edges = jumps in the control flow
- Every block: starts from jump target, ends by jump command

Control Flow: Basic Tricks

Any ideas for control flow obfuscation?

- Break basic blocks
- Inline methods
- Outline statements
- Unroll loops
- Reorder statements
- Reorder loops

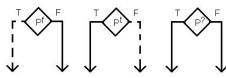
How to Destroy a Control Flow Graph?



- 1 Write down a list of all basic blocks
- 2 Split and merge some of them
- 3 Enumerate them
- 4 Replace all calls by indirect pointing
- 5 Write a single dispatcher to maintain all control flow

Opaque Predicates

How can we use IF operator for obfuscation?



Opaque predicates: every time the same value
Difficult to discover by automatic static analysis

Examples:

$$((q + q^2) \bmod 2) = 0$$

$$((q^4) \bmod 16) = 0 \text{ OR } ((q^4) \bmod 16) = 1$$

Functions Unifying

How can we make program procedures indistinguishable?

Idea:

- Merge all functions to one
- Call universal function with additional parameter

Difficulty: different **signatures** (input-output specifications)

Solution: unify signatures (in groups)

Even more transformations

Question: Can you invent more?

- Reuse identifiers
- Introduce misleading comments :-)
- Modify inheritance relations
- Convert static data to procedural data
- Store part of the program as a text and interpret it only during runtime
- Remove library calls
- Protection against specific decompiling tools

Current Techniques: Pro and Contra

Advantages:

- ✓ Easy to implement
- ✓ Universal
- ✓ Good against static analysis

Disadvantages:

- ✗ No guaranteed security
- ✗ Even no hope for that
- ✗ Weak against dynamic attacks

Summary

Main points:

- Obfuscator workflow: parse the program; apply transformations until the cost is exceeded
- Obfuscating transformations consist of **layout, data and control** tricks
- Hardness of deobfuscation is **not proved**






Course Conclusion

Why programming people like code obfuscation so much?

Programming: CONSTRUCTIVE process

Obfuscation: DESTRUCTIVE process

Reading List

-  [C. Collberg, C. Thomborson, D. Low](#)
A taxonomy of obfuscating transformations, 1997.
<http://www.cs.arizona.edu/people/collberg/Research/Publications/CollbergThomborsonLow97a/A4.ps>
-  [C. Collberg, C. Thomborson, D. Low](#)
Breaking abstractions and unstructuring data structures, 1998.
<http://www.cs.arizona.edu/~collberg/Research/Publications/CollbergThomborsonLow98b/LETTER.ps>
-  [S. Chow, Y. Gu, H. Johnson, V. Zakharov](#)
An approach to the obfuscation of control-flow of sequential computer programs, 1998.
<http://www.ispras.ru/groups/dma/downloads/Malaga2.zip>
-  [M. Mambo, T. Murayama, E. Okamoto](#)
A tentative approach to constructing tamper-resistant software, 1998.
<http://web.yl.is.s.u-tokyo.ac.jp/~cocoa/reading/archive/p23-mambo.pdf>
-  [C. Linn, S. Debray](#)
Obfuscation of executable code to improve resistance to static disassembly, 2003.
<http://www.cs.arizona.edu/~linnc/research/CCS2003.pdf>

Thanks for attention. **Questions?**

Course Feedback

- 1 Comments/suggestions on **contents**:
 - Choice of topics? Ratio of theoretical/practical?
- 2 Comments/suggestions on **presentation aspects**:
 - Your opinion on slides? Black board explanation? Language mistakes?
- 3 Comments/suggestions on **technical aspects**:
 - Timetable of the course? Webpage? Room? Announcement?
- 4 Main **advantage** of the course (if any)?
 - Best lecture in your opinion?
- 5 Disatvantages. **What** and how **can be improved**?