Introduction to Program Obfuscation

Yury Lifshits
Saint-Petersburg State University

http://logic.pdmi.ras.ru/~yura/
yura@logic.pdmi.ras.ru

Some Linguistics

Obfuscate: tr.v. -cated, -cating, -cates. 1. a. To render obscure. b. To darken. 2. To confuse: his emotions obfuscated his judgment. [Lat. obfuscare, to darken : ob(intensive) + Lat. fuscare, to darken < fuscus, dark.] -obfuscation n. obfuscatory adj

---

In German **Obfuscation** is

die Benebelung

die Trübung

die Verdunkelung

die Verschleierung - von Tatsachen

die Verwirrung

Overview

- Introduction:
  Notion of obfuscation
  Motivation and applications
  Research history and commercial obfuscators
- State-of-the-art:
  Code tricks
  Theoretical approach & provable security
- Conclusions
  Evaluation of current results and future research

Notion of Obfuscation

An **obfuscator**: An algorithm O such that for any program P, $O(P)$ is also a program with following properties:

- Functionality: The obfuscated program should have the same functionality (that is, input/output behavior) as the input program.
- Efficiency: The obfuscated program shouldn't be much less efficient than the input program.
- Obfuscation: This means that the code of the obfuscated program should be hard to understand.

Applications

- Protection of constants and data of the program

  Authentication schemes, e-money, license management

- Protection from intelligent tampering

  E-money, license management

- Algorithms Protection

  Defence against competitors

- Viruses modification

  Making old viruses unrecognizable

- Private key cryptosystems $\rightarrow$ Public key cryptosystem

  Basic idea: public key = obfuscated encrypting algorithm of private key cryptosystem

Motivation to research

- Practical necessity

  Wide use of Java byte-code technology

- New topic: not well developed yet

  No single generally accepted formal defi nition of obfuscation exists

- We hope: there are good obfuscation algorithms

  Average code is very obfuscated; Rice's theorem; Classically hard problems

- Famous researchers and institutes are involved

  Weizmann, Princeton, Stanford; O. Goldreich, P.C. van Oorschot

- The International Obfuscated C Code Contest

  `http://ioccc.org`

- Can be approached both in theoretical & practical ways

Research Highlights

1997 A taxonomy of obfuscating transformations – C. Collberg, C. Thomborson, D. Low

2001 Impossibility result – B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang.

2003 First attempt to survey – L. D'Anna, B. Matt, A. Reisse, T. Van Vleck, S. Schwab and P. LeBlanc.

A series of Ph.D Dissertations: G. Wroblewski, D. Low, C. Wang . . .
$\Longrightarrow$ Good Ph.D. topic!

Commercial Obfuscators

- Most common techniques:
  - ✓ name mangling
  - ✓ control flow mangling
  - ✓ strings encryption
- SandMark

  `www.cs.arizona.edu/sandmark/`
- Cloakware, Retroguard, DashO, Klassmaster, yGuard & many more . . .

`http://dmoz.org/` → Computers → Programming → Languages → Java → Development Tools → Obfuscators/

State-of-the-Art

- Code transformations
  Pro et Contra
  Basic tricks
  Opaque predicates
  Flat control flow
- Theoretic approach
  Pro et Contra
  Blackbox security
  Examples of obfuscation with cryptographic security
- Other research directions

Coding transformations

- Advantages
  Easy to implement
  Universal
  Good against static analysis
- Disadvantages
  No guaranteed security
  Even no hope for that
  Weak against dynamic attacks

Simple tricks

- Split & merge variables, constants, procedures, modules
- Increase & decrease dimension of arrays
- Increase & decrease nesting
- Addressing & dereferencing
- Renaming
- Reordering
- Cloning
- Strings encrypting

Opaque predicates

- Reordering of blocks execution
- Dead code insertion
- Inserting new `IF` operators

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

`If` $((q+q^2)\ \ mod\ \ 2)\ \ =\ \ 0$ `then do` *real work* `else do` *dead code*

`If (`*any boolean expression*`) then do` *real work* `else do` *just the same*

Control flow flattening

- Write down a list of all basic blocks
- Split and merge some of them
- Enumerate them
- Replace all calls by indirect pointing:

  `goto` *block_name*

  `goto` *block_number*`-th block`

  `goto` v`-th block`
- Write a single dispatcher to maintain all control flow

Provable security

- Good News

  $+$ Guaranteed security!

  based on computationally hard problems

  $+$ Some positive results

- Bad News

  – Now: only protection of internal constants.

  P computes $f(x, p)$. Task: protect p.

  – No hope for universal method

Black-box security

Informally: an obfuscator should provide a **virtual black-box** in the sense that giving a $O(P)$ code to someone is equivalent to giving him a black box that computes P

Just the same: anything that can be learned from the obfuscated form, could have been learned by merely observing the programs input-output behavior (i.e., by treating the program as a black-box).

This definition is impossible to meet!

Interactive access control

- Directed multi-graph G
- Each node representing an access point
  (some abstract secrets & local map inside)
- Each edge has a password checking on it
- S is predefined start access point (start node)
- User: knows some passwords
  No a priori knowledge about G
  Begins his way from S

IAC task for obfuscation

- The user can reach an access point only by presenting credentials that can take him from the start node to that point.
- The user gains complete access to a function or secret available at an access point if and only if the user has reached that access point.
- The user does not learn anything about the structure of the graph, except what is revealed by the secrets at the access points he reached and the edges he traversed.

Result[2004]: Black-box security achieved!

Security based on (existence of) pseudorandom functions

Hiding password checking

Program Π:

```
var x:string, y:bit;
input(x);
y:=0; output(y);
```

Family of Programs Π_k:

```
var x:string, y:bit;
input(x);
if k = w then y := 1 else y := 0;
output(y);
```

Task: make these programs indistinguishable

Result[2001]: Any probabilistic polynomial algorithm can recognize the actual case with at most $1/2 + neg(size\ of\ pass)$ probability.

Based on (existing of) one-way permutations.

Not in this talk

- Secure architectures approach

  new presentation forms to distribute programs

- Obfuscation in multiparty systems

  splitting program to the set of communicating programs

- Making disassembling harder

Conclusions

- Evaluation of the current results
- Important research directions
- Some useful links

Has been already done

- Many coding transformations
- Obscuring static analysis
- Some obfuscations with cryptographic security
- First steps from general method to attack-dependent obfuscation

Necessary to do

- Measuring quality of obfuscation algorithms
  Evaluating of existing methods

  Now: only code complexity metrics
- Study of deobfuscation algorithms

  Finding hard problems for code analysis
- Universal mathematical model
- Cryptographic (computational) security for obfuscation

Best links to start with

- L. D'Anna, B. Matt, A. Reisse, T. Van Vleck, S. Schwab and P. LeBlanc. Self-protecting mobile agents obfuscation report. Technical Report #03-015, Network Associates Labs, June 2003:
  `http://opensource.nailabs.com/jbet/papers/obfreport.pdf`
- "Code Obfuscation" presentation by Mayur Kamat & Nishant Kumat:
  `http://ee.tamu.edu/˜reddy/ee689_04/pres_mayur_nishant.pdf`
- Obfuscators subdirectory of Dmoz.org:
  `http://dmoz.org/Computers/Programming/Languages/Java/Development_Tools/Obfuscators/`
- Resources on Obfuscation (huge collection of links to related papers, books & companies):
  `http://www.scs.carleton.ca/˜hshen2/` (resources section)

The End

Thanks for your attention!

Question Time

? ? ? ? ? ? ?

Not covered by the talk

Program Obfuscation as a part of Software Protection
Operations on obfuscated code
Micro-obfuscation: functions, procedures, data structures
Adversary knowledge about the program
Nonfunctional models of a program.
Cost of the obfuscation
Potential of obfuscation
What can obfuscation change in the program?
Different behavior on different runs (internal memory of the program).
Nondeterministic nature of the obfuscator
Effi ciency of obfuscating transformations
Obfuscatable program properties (e.g. set of all possible output values)
Obfuscation by hiding small procedure in the big one (in steganographic style)