Open Problems in Program Obfuscation

Yury Lifshits

Mathematics & Mechanics Faculty
Saint Petersburg State University

Spring 2005 – SETLab

Outline

1 Practical Approach

- More Transformations
- Obfuscation Benchmarks
- Obfuscating Language

Outline

1 Practical Approach

- More Transformations
- Obfuscation Benchmarks
- Obfuscating Language

2 Central Tasks

- Integrity Protection
- Data Structures Obfuscation
- Outside the Standard Model

Outline

1. **Practical Approach**
   - More Transformations
   - Obfuscation Benchmarks
   - Obfuscating Language
2. **Central Tasks**
   - Integrity Protection
   - Data Structures Obfuscation
   - Outside the Standard Model
3. **Approaches to Obfuscation**
   - Deobfuscation Research
   - Cryptography & Obfuscation
   - Random Program Idea

More transformations

What are promising ideas you can invent?

More transformations

What are promising ideas you can invent?

- Traces obfuscation
- Making Flow Graph strongly non-reducible
- Preventive code transformations
- Security against dynamic attacks
- Protection against slicing

More transformations

Slide from Lecture 1 — your turn to explain.

More transformations

Slide from Lecture 1 — your turn to explain.

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

Example:

$$\boxed{\boxed{((q+q^2)\ mod\ 2)\ =\ 0}}$$

More transformations

Slide from Lecture 1 — your turn to explain.

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

Example:

$$\boxed{\boxed{((q+q^2)\ mod\ 2)\ =\ 0}}$$

Research task: to generalize this idea to opaque states.
Study theoretical power of this idea.

Obfuscating fixed program

Let us fixed some program P. Then we can ask for the best obfuscation of P.

- Contest for the best obfuscation
- Challenge contest for deobfuscation

We can compare different obfuscators studying their results on one test program.

Another idea: take two programs P_1 and P_2, which are difficult to distinguish by black-box testing. Then check whether it is possible to distinguish their obfuscated versions.

Obfuscating Language

Observation: many of obfuscating transformations looks like translation to another artificial programming language.

Some properties of this “obfuscating language”:

- No high-level constructions
- Low modularity, high interdependency
- Reusing identifiers
- Wide usage of pointers

Research task: to understand utility of constructing such a language.

Informal Concept of Integrity

So, what is integrity and integrity protection?

Informal Concept of Integrity

So, what is integrity and integrity protection?

Informal concept:

- Fixed order of computation
- Undetachability
- Protection of IF operator
- Tamper resistance

Applications of Integrity Protection

Once again, what are applications of integrity protection?

Applications of Integrity Protection

Once again, what are applications of integrity protection?

- ⇨ Watermarking
- ⇨ Delegating restricted authority (in mobile agents)
- ⇨ Bounded functionality
- ⇨ Competitor threat
- ⇨ Protection of licence management & password checking schemes

Informally about Data Obfuscation

So, what is data protection and obfuscation?

Informally about Data Obfuscation

So, what is data protection and obfuscation?

Informal concept:

- ⇨ Difficulty of changes with predicted effect
- ⇨ Intermediate results are meaningless (or encoded)
- ⇨ Important constants are never kept in decrypted form even during runtime
- ⇨ Every data item seems to be similar to every other one

Applications of Integrity Protection

Once again, what are applications of data protection?

Applications of Integrity Protection

Once again, what are applications of data protection?

- ⇨ Mobile agent state protection
- ⇨ Keys hiding
- ⇨ Again, tamper resistance
- ⇨ Again, protection of licence management & password checking schemes

Guaranteed Slowdown

Why might we be interested in slowdown of programs?

Guaranteed Slowdown

Why might we be interested in slowdown of programs?

To protect cryptosystems against brute force attacks!

Obfuscation task: To compile program P into program $O(P)$ with the same functionality and such that:

- ⇨ $O(P)$ works essentially slower than P does
- ⇨ Given O(P) it is (computationally) difficult to make speedup back to the level of P

Market-Based Parallel Computing

Informally: We have some difficult computational problem divided to many work packages. Wy want to buy computational resources to run this packages and bring back results. Finally there is a security requirement:

- ⇨ We want to guarantee that during this computation nobody gain any information about our original task and involved data.

Difference with ordinary obfuscation: computers running our packages not need to produce clear (decrypted) results. So this task seems easier than others.

Deobfuscation Research

What are interesting questions about deobfuscation?

Deobfuscation Research

What are interesting questions about deobfuscation?

General idea: make current deobfuscation methods inefficient or producing meaningless results.

Research tasks:

- Write down top ten deobfuscation tricks
- Find and study hard problems in program analysis
- Find and destroy invariants of current code transformations
- Build deobfuscation instruments classification

Cryptography & Obfuscation

Obfuscation for cryptography. **Research tasks**:

- ⇨ Construct homomorphic encryption schemes based on obfuscation
- ⇨ Construct function computation with protection against inversion (similar to “private→public” application)

Cryptography & Obfuscation

Obfuscation for cryptography. **Research tasks**:

- ⇨ Construct homomorphic encryption schemes based on obfuscation
- ⇨ Construct function computation with protection against inversion (similar to "private→public" application)

Cryptography for obfuscation. **Research tasks**:

- ⇨ Find a reasonable class of functions with possible black-box secure obfuscation
- ⇨ Find a reasonable class of programs with possible efficient encrypted computation schemes
- ⇨ Find more utilizations and connections between classical cryptography and software protection

Random Program Idea

⇨ Let us fix program P we want to obfuscate

Random Program Idea

- Let us fix program *P* we want to obfuscate
- Then let us fix our obfuscation constraints (time, space, code size)

Random Program Idea

- ⇨ Let us fix program P we want to obfuscate
- ⇨ Then let us fix our obfuscation constraints (time, space, code size)
- ⇨ Now we can define obfuscation set S as a set of all programs having the same functionality as P has and satisfying all constraints

Random Program Idea

- ⇨ Let us fix program *P* we want to obfuscate
- ⇨ Then let us fix our obfuscation constraints (time, space, code size)
- ⇨ Now we can define obfuscation set *S* as a set of all programs having the same functionality as *P* has and satisfying all constraints

Random Program Idea

- ⇨ Let us fix program P we want to obfuscate
- ⇨ Then let us fix our obfuscation constraints (time, space, code size)
- ⇨ Now we can define obfuscation set S as a set of all programs having the same functionality as P has and satisfying all constraints

Best solution: take the most “unreadable” representative of S. Two difficulties: we still have no strict definition of “unreadable” and this way seems to be very hard to implement.

Random program idea: Assume that we can construct a random representative of S class. There is a hope that w.h.p. this program would be much more difficult to analyse than P and hardness of analysis would be quite close to the worst representative case.

Summary

- We need some quality measurement for practical approach. Possible way out is introducing benchmarks and starting challenge contests.
- For obfuscation against fixed attack the most important case is integrity protection.
- There is hope for wide use of cryptographic primitives in obfuscation.

Summary

- We need some quality measurement for practical approach. Possible way out is introducing benchmarks and starting challenge contests.
- For obfuscation against fixed attack the most important case is integrity protection.
- There is hope for wide use of cryptographic primitives in obfuscation.

Question Time!

Not Covered by the Talk

Obfuscating of key generator algorithm

Quality = task complete

Smart card

Properties / algorithm hiding

Obfuscation primitives

Micro-obfuscation

Models of communication?

Black-box reverse engineering

Inductive constructions

Not Covered by the Course

Disassembling
JVM obfuscation
DES obfuscation
Obfuscator evaluation and comparison.
Deobfuscation and hacker tricks