(A brief introduction to)
Program obfuscation

Anton Lokhmotov
University of Cambridge Computer Laboratory
http://www.cl.cam.ac.uk/users/al407/

Joint work with N.P. Varnovsky and V.A. Zakharov
Institute for Systems Programming RAS
http://www.ispras.ru/groups/dma/dma.html

Spring School on Security, Marseille, France
27th April, 2005
Agenda

• Introduction.
• What is obfuscation?
• Why to obfuscate?
• (In)security of obfuscation.
• (Im)possibility of obfuscation.
• An attempt to build a secure obfuscator.
• Conclusion.
What is obfuscation?

By program obfuscation we mean an efficient transformation of computer programs aimed at rendering programs “unintelligible” in some sense, while preserving their functionality.

**(Informal) definition** [Barak *et al*]
An obfuscator $O$ is an (efficient, probabilistic) “compiler” that takes as input a program $P$ and produces a new program $O(P)$ satisfying the following two conditions:

- (functionality) $O(P)$ computes the same function as $P$;
- (“virtual black box” property) “Anything that can be efficiently computed from $O(P)$ can be efficiently computed given access only to inputs and outputs of $P$.”
Why to obfuscate?

Well, some people just do it for fun!

International Obfuscated C Code Contest (http://www.ioccc.org)

```c
#include <stdio.h>
int l; int main(int o, char **O, int I) {char c, *D = O[1]; if (o > 0) {
    for (l = 0; D[l] ; D[l++] -= 10) {D[l] = (D[l] + 82) % 10 - (I > l / 2) * (D[l] + 72) / 10 - 9;
        D[I] += I < 0 ? 0 : !(o = main(c / 10, O, I - 1)) * ((c + 999) % 10 - (D[I] + 92) % 10);}
    return o;
}
```

Best short program ’2001 by Raymond Cheong, Johns Hopkins University, USA

Judges’ comment: “The source code is nice, compact, and self documenting as all good programs should be! :-)”
Why to obfuscate?

- Program obfuscation is generally conceived as a form of software protection against *malicious reverse engineering* - unauthorised extraction of valuable information (e.g., algorithms or data) from the code. It also can help to hide *vulnerabilities* and prevent *program tampering*.

- Obfuscation is particularly important for the code distributed in hardware-independent formats such as the Java Virtual Machine and the .NET Common Language Runtime, which retain a lot of information about the source program and thus are relatively easy to reverse engineer.
(In)security of obfuscation

- Most known methods of obfuscation are just the reverse of good software development practice;
- none of them guarantees that it is impossible to retrieve sensitive information about algorithms and data structures from the obfuscated program;
- this is exemplified by successful attempts on deobfuscation (e.g. [NAL]), which testify to a widespread (and judicious) opinion that:
  - a determined specialist, given enough time and resources, is able to deobfuscate any obfuscated program.

(Im)possibility of obfuscation

Definition [Barak et al]
A probabilistic algorithm $\mathcal{O}$ it a TM obfuscator if the following holds:

- For every TM $M$ any output $\mathcal{O}(M)$ of $\mathcal{O}$ on input $M$ describes a TM that computes the same function as $M$.

- The description length and running time of any TM $\mathcal{O}(M)$ are at most polynomially larger than that of $M$. I.e., there exists a polynomial $p$ such that for every TM $M$ and any $\mathcal{O}(M)$, $|\mathcal{O}(M)| \leq p(|M|)$ and if $M$ halts in $t$ steps on some input $x$ then $\mathcal{O}(M)$ halts within $p(t)$ steps on $x$.

- For any PPT $A$ there is a PPT $S$ and a negligible function $\nu$ such that for all TMs $M$

\[
|Pr\{A(\mathcal{O}(M)) = 1\} - Pr\{S^M(1^{\|M\|}) = 1\}| = \nu(|M|).
\]
(Im)possibility of obfuscation (cont’d)

Theorem (basic impossibility result) [Barak el al]
Turing machine obfuscators (in the sense of the above definition) do not exist.

However, this result does not imply that there is no method of making programs “unintelligible” in some meaningful and precise way. Thus, it is both important and interesting to understand whether there are alternative senses (or models) in which some form of obfuscation is possible and useful for software protection.

An attempt to build a secure obfuscator

Our aim is

- to develop a practicable obfuscation method for programs written in a variety of imperative programming languages, and
- to prove its security against some threats and attacks.

We developed

- an obfuscating (intra-procedural) control flow transformation
- for programs written in high-level imperative programming languages (like C),
- using cryptographic primitives (a pseudorandom generator and a hash function, both based on the subset-sum problem),
- which we believe is (cryptographically) secure against (brute force) static control flow analysis (still needs a formal proof :-).
• Program obfuscation is a form of software protection, which has implications for security, \textit{but}

• most (nearly all?) currently available obfuscation practices provide no guaranteed protection, \textit{so}

• please, bring new ideas for obfuscation, including sensible definitions and proofs of security!