Towards Certified Program Obfuscation

Weiyun Lu
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
WLU058@uottawa.ca

Bahman Sistany
Cloakware Research
Irdeto Canada
Ottawa, Canada
bahman.sistany@irdeto.com

Amy Felty
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
afelty@uottawa.ca

Philip Scott
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
philip.scott@uottawa.ca

***Abstract*—We expect our systems including software systems to function "correctly". By "correctly", we mean that a system will behave according to explicit and/or implicit expectations or said another way to its written and/or unwritten specifications. Typically, extensive testing is done to increase the confidence in correct functionality of a piece of software but positive test results are not a proof of correctness. In systems where more assurance of correctness is required, formal verification based methods are used. Behaviour of interest such as functionality, safety and security may be expressed via some sort of formal specification language.**

How can one perform code transformations such as obfuscating transformations or optimizing transformations on code that is assumed to be correct wrt to certain specified behaviour? Will the transformed code preserve the specified behaviour as one expects?

To achieve the highest levels of assurance that the transformations have maintained correctness of the code, is to prove the two versions of the program (before and after the transformation) is equivalent. Total equivalency between the two versions of a program certainly implies the correctness of any specified properties of interest but what if we directly try to show validity of these properties on the transformed program?

In this thesis, we lay the foundation to study and reason about code obfuscating transformations and show how the preservation of certain behaviours may be "certified". To this end, we apply techniques of formal specification and verification, by using the Coq Proof Assistant and IMP (a simple imperative language within it), to formulate what it means for a program's semantics to be preserved by an obfuscating transformation, and give formal machine-checked proofs that these properties hold.

We describe our work on opaque predicates, a simple control flow obfuscation, and elements of control flow flattening transformation, a more complex control flow obfuscation. Along the way, we employ Hoare logic as our foundational specification language, as well as augment the IMP language with Switch statements. We also define a lower-level flowchart language to wrap around IMP for modelling certain flattening transformations, treating blocks of codes as objects in their own right.

***Index Terms*—obfuscation, verification, security, correctness, Coq, proof**

1. Introduction

1.1. Background and Motivation

1.2. Summary and outline of the rest of the paper