Program Obfuscation
April 24th, 2014

Summary and background:

This paper is about limiting what an attacker can learn about a program via reverse engineering. The authors explain that if the fixed program semantics of a program are complete than an attacker's analysis may be exact. That is to say that an attacker's abstract interpretation of the program will be precise. They then go on to say that if they can force a program's semantics to be incomplete, the attacker's analysis will be imprecise and in this sense they will have learned less about the program than they would have on a non-obfuscated version. The trick in the paper is transforming the input program in specific ways to directly control which parts of the abstract semantics are incomplete and thus control specifically what an attacker may learn about a program.

-for instance:

An abstract interpretation of a program which implements multiplication and gives its answer in the domain of signs might be complete because the abstract interpretation of the operators for multiplication are precise. However, if we implement an obfuscated version which implements multiplication using addition (where the domain of '+' is not complete) then the abstract interpretation will be incomplete as well because the abstract interpretation is imprecise in the domain of signs for addition operations

Critiques:

This paper was "obfuscated" and poorly written. -class as a whole

Questions and Discussion:

Mike: Why should we care? (Sympathetic Use Case)

Kris: Intellectual Property

Literally obfuscate code to prevent people from literally stealing code

Matt: Obfuscate code so that a non-client controlled computer can't detect what program the client is running

Andrew: Obfuscate Patches

Prevent bad-guys from reverse engineering vuln-patches

Mike: You've successfullly obfuscated your program if...?

Jonathan: An adversary can only learn as much as a simulator that only has access to inputs an ouputs

Mike: That definition has an impossibility result. What are other flaws in the impossibility result? The potential advantage of abstract interpretation is that this definition of complete obfuscation is very strong. Using abstract interpretation lets us not care about the black box interpretation but allows us to specify specific properties about the program which we don't want the adversary to know without caring about the fact that the adversary can still learn somethings.

Mike: Suppose you had a program which runs on a distribution of a programs with a given input and produces a distribution of outputs. If you view the input distribution as an adversary's beliefs, than there is an information flow because the adversary can reduce the belief distribution from the input distribution to the distribution of only those programs whose output is contained in the overall output.

Aseem and Mike:

In a way, adding sleeps to programs to deter timing analysis (ala the side channel papers from earlier) is analogous to program obfuscation

Mike: What does it mean to understand a program?

James: One meaning could be "find a secret key"

Aseem: Maybe I want to obbfuscate to give to a cloud provider to other cloud users can't steal my code

Upper Closure Operators: function that is monotone, extensive and idempotent.

These are crucial to abstract interpretation and obfuscation but the precise details are complicated

GLB of (UCO s.t. semantic properties are preserved) = most concrete property preserved