Thesis

Gilles De Borger

March 31, 2021

Dag iedereen,
vandaag ga ik jullie vertellen over mijn thesis.

1 Introduction

I am writing a tool to mitigate Nemesis attacks. To mitigate attacks a specific condition has to hold, that for any i, the i'th instruction in the if and in the else branch have the same latency. In order to ensure this property holds I instrument the binary code with new instructions to ensure that this property holds for all secret dependent branches without modifying the program execution (i.e. all instruction should have no effect)

2 Implementation (scratch)

2.1 control flow graph

The implementation consists of a number of algorithms and operations that are performed on the program's control flow graph (CFG). The CFG is a graph with nodes V and edges E such that each node in V represents a sequence of instructions without any branching, and each edge E represents jumps between nodes. It is possible to first create a CFG given a program and later reconstruct the program using the (modified) CFG.

Two nodes of the CFG are said to be balanced if they have the same number of instructions and if each corresponding instruction has the same latency. A node satisfies the Nemesis property if for any descendants at depth i it holds that the nodes are balanced.

Ensuring the Nemesis property holds for a given node is relatively straightforward under certain conditions. If the CFG has a tree structure a recursive strategy is possible to ensure the Nemesis property holds for a given node. Unfortunately a programs control flow rarely has a tree structure. To

3 Questions

When I refer to Control Flow Graph I do not mean the 'general' control flow graph. I mean an actual datastructure that I use. Should I make this more clear?