More faith in formal isolation properties

In our group at KTH, we study improvements of security properties for execution platform software. The project funding my PhD studies aims at critical infrastructure applications such as smart grids and traffic control systems. These systems become more and more connected to the internet and thus need to be made more resilient against attacks. The project is made up of four areas, only one of which our group is responsible for. We are applying formal verification for improving security in terms of execution domain isolation in hypervisor software. The other three are researching resilient control algorithms, secure communication protocols and physical layer network security.

Since we apply software verification at binary level, processor models are needed. As an example, we study ARM architectures. Previous research in our group took existing models and extended them, e.g. with virtual memory by adding a suitable model of the memory management unit (MMU).

The proof of hypervisor isolation is comprised of different parts. Firstly, a user lemma proving that user-mode instructions do not affect hypervisor state. Secondly, the definition of an MMU page table configuration invariant. This states that valid page tables do not allow the execution domains to affect page table contents. Thirdly, we split hypervisor code in two parts, i.e. the page table and MMU related part and the rest. For the former we modelled the hypervisor software in our proving system (i.e. HOL4).

Part of the proving tool chain consists of a lifter: a software that creates from binary code semantically equivalent machine independent intermediate code (in the BIL language). Another tool, called binary analysis program (BAP), is used to generate the weakest precondition of a BIL program for further analysis. In order to improve faith in the proof, we want to integrate the Lifter into the proof system and obtain a proof for the lifting being correct. This is achieved by a proof-producing compilation.

The new Lifter produces a proof that demonstrates a simulation between the original ARM program and the produced BIL program. This theorem allows to transfer properties verified for the BIL program to the ARM one.

In the future we plan to experiment with generating code directly from our software models of the page table and MMU related code. We want to use an intermediate language of the proof-producing CakeML system. Then, together with a proof-producing code generation from our software models, we would have a trustworthy compilation, which does not require lifting or any further effort to prove semantic equivalence.