Isolation of a Network Interface Controller

Jonas Haglund
jhagl@kth.se, KTH Royal Institute of Technology

Formally verified execution platforms (e.g. hypervisors) provide an infrastructure for implementing secure IoT devices. By guaranteeing memory isolation and controlling communication between software components, they prevent faults of non-critical software (e.g. large and complex commodity software) from affecting security or safety critical software. This enables formal verification of critical software without considering non-critical software.

An issue with some of these verified execution platforms is that I/O devices are not considered. Either I/O devices are disabled (making the IoT device useless), or device drivers (whose code is comparable in complexity and size to the verified execution platform) are trusted. In the latter case isolation could be violated if a driver abuses Direct Memory Access (DMA) of an I/O device.

We designed an IoT device whose security relies on isolation and the principle of complete mediation: The device driver of the Network Interface Controller (NIC) is part of untrusted software and its NIC reconfigurations are checked by a run-time monitor to preserve a security policy, phrased as an invariant. The invariant is preserved by the NIC and implies that the NIC cannot access certain memory locations. The security of this design depends on: (1) Correct isolation of the monitor; (2) the monitor is correct by denying NIC reconfigurations that violate the invariant; and (3) the invariant ensures that the NIC does not break isolation.

We accomplished the first formal verification of isolation (3) for a real hardware I/O device of significant complexity: The NIC of Beaglebone Black. We formalized the model of the NIC, and we defined the security policy as an invariant of its state. Then we proved in the HOL4 interactive theorem prover that the invariant is preserved by the NIC and that it restricts DMA accesses to only certain memory locations.

The software of the demonstrator consists of a secure hypervisor that hosts Linux, a secure service, and the monitor. Linux controls the NIC, but the monitor, implemented in the hypervisor, intercepts each NIC reconfiguration and checks that it does not enable the NIC to access memory not part of Linux. This enables the secure service to access the Internet, using Linux as an untrusted intermediary.