Research Institute
Cyber Defence
Bundeswehr University Munich

Annual Conference CODE 2020 - Summary of Workshop 4

Cyber Security Through Formal Software Verification

As part of CODE 2020 a workshop on the use of formal software verification for achieving cyber security was organized by Gunnar Teege, Bundeswehr University Munich. The goal of the workshop was to present the current state of the subject and discuss whether it is ready for use in practical applications. The workshop consisted of four presentations by invited speakers and a panel.

Due to the Covid-19 pandemie the workshop, like the whole CODE 2020 conference, was held completely in virtual form. For the audience it was possible to switch freely between the 7 parallel workshops, in workshop 4 there were about 15 – 25 virtual visitors present.

The first presentation by Tobias Nipkow (Technical University Munich) was an introduction to basic tools with the title „Interactive Theorem Provers: What they are and what they can do“. This talk addressed the following questions:

- What are interactive theorem provers?
- What do such proofs look like?
- How are they construced?
- What landmark proofs have been achieved?
- What is the impact on the academic community?

In particular, it presented the interactive theorem prover Isabelle and explained the difference between automatic proof generation and manual („interactive“) proof generation.

The next talk by Gernot Heiser (University of New South Wales and CSIRO / Data61, Australia) presented the seL4 project and some of the experiences made with formal software verification using Isabelle. Titled „seL4: Verified Operating System for the Real World“ the talk gave an overview what seL4 is and what its verification does (and does not) mean in practice. It also discussed how seL4's proved isolation enforcement can help to protect real-world systems from cyber attacks, and presented concrete deployment scenarios. Until about 10 years ago, formal verification of software was mostly considered an academic exercise that doesn't scale to real-world problems. seL4 changed that: it is not only the world's first operating system (OS) kernel that is verified to the code level, it was, from the beginning, designed to be deployed in real-world security- and safety-critical systems.

In his talk titled „Practical Verification for Software Engineers“ Alexander Senier (Componolit, Dresden) presented experiences with SPARK and the RecordFlux toolset. Often, the verification of existing software is infeasible and a complete rewrite can be prohibitively expensive. SPARK offers different levels of assurance with different costs and benefits. By replacing critical parts of an existing software by verified code, security can be strengthened significantly with moderate effort. By using high-level abstractions and

automation the required effort can be even further reduced. By automatic generation of SPARK code the RecordFlux toolset allows proving various properties of a specified protocol and ensures absence of runtime errors and functional correctness of the generated code.

The final presentation by Jaap Boender (Hensoldt Cyber, Taufkirchen) had the title „Hands on: Watching software being verified“. In this talk, he looked at deductive verification of software from a more practical point of view. After a short introduction, he took a simple program and used it as a running example to show the different techniques of verification, and discussed their applications, their advantages and disadvantages. The talk remained fairly high-level, and accessible to non-experts. It was intended to give an insight into the possibilities that are offered by deductive software verification, and how it can be applied to larger-scale software projects. Nevertheless the presentation gave a clear impression how difficult and complex interactive proofs of software security properties still are.

In the panel the following additional findings were addressed by the workshop participants:

- Hardware properties can still be a problem, if they are not fully included in the proven software behavior. The current development in the field of RISC V processors tries to mitigate this by making all microarchitectural state explicit, so that in can be included in proofs.
- Automatic proof tools are necessary to reduce the effort and costs of verification, however, at the current state they are not sufficient and must still be complemented by manual proof work for a complete proof of security properties in many cases.
- The number of existing specialists who can do interactive formal software verification is estimated to be only some thousands in the world. Systematic qualification programs are rare.
- Often, to reduce costs, only the most critical parts of a system are formally verified. This must be done with care, strictly isolating them from the rest of the system (and verifying the isolation). Otherwise a false perception of security can result, where weaknesses in the rest of the system can be used to compromise the verified parts.
- Currently, the costs of formal verification are still too high for normal applications. But for security critical applications (of all kinds) the effort pays off, by providing better security than other approaches with similar costs.