Reliability and Availability Working Group

Minutes for the RAWG meeting on 9th December 2024

Present: A. Arias Vazquez, K. Blantos, M. Blaskiewicz, H. Boukabache, E. Boulharts, K. Ceesay-Seitz, S. Er, B. Fernandez Adiego, X. Fink, Q. Genoud, B. Guncic, P. Jurcso, F. Kolitsi, A. Kostic, S. Kulis, T. Ladzinski, C. Laforge, H. Leicester, I. Lopez-Miguel (TU Wien), M. Lupi, T. Miceli, D. Perrin, A. Pinho, A. Pulli, M. Saccani, S. Scarfi, N. Voumard, F. Waldhauser, D. Westermann, L. Zwalinski

RAWG Meeting: Formal Methods and Verification

The slides are available on indico.

Agenda

L. Felsberger opened the meeting focused on formal methods and formal verification. He welcomed **K. Ceesay-Seitz**, a former CERN fellow working on formal verification within the CROME project (HSE-RP). **K. Ceesay-Seitz** is now a PhD student at ETH Zurich and will give an update on her latest research. Moreover, **X. Fink** will present the latest PLCVerif extension. **L. Felsberger** noted that activities in this field at CERN are consolidated under the Formal Methods Interest Group. He also mentioned a questionnaire focused on the development of reliable and robust gateware, which has been distributed to relevant equipment groups.

1 µCFI: Formal Verification of Microarchitectural Control-flow Integrity - K. Ceesay-Seitz

K. Ceesay-Seitz presented µCFI, a generalized security property for formal verification of the microarchitectural control-flow integrity. It is independent of the CPU's verification state and can thus be applied early in the design cycle. The goal is to prove that no timing side channels exist, which could be used for hijacking. CellIFT, a previous method, tracks all the information flows with taint logic. CellDFT, a variant of CellIFT, tracks only data flows coming from the input. µCFI combines CellIFT with formal verification to prove that the operand data cannot influence the program counter. The proposed approach shows an automated verification method and implementation, which allowed discovery of five new vulnerabilities during verification of four open-source RISC-V CPUs.

Discussion after the presentation:

H. Boukabache asked if Cadence Jasper is used for implementation and if the approach would be different from Siemens Questa. **K. Ceesay-Seitz** confirmed that Cadence Jasper is used for implementation and that the approach is similar to Questa. **H. Boukabache** further asked if **K. Ceesay-Seitz** is working on the hardware implementation or hardware description level and what the differences between the evaluated processors are. **K. Ceesay-Seitz** replied that she is working on hardware implementation and that the evaluated CPUs are different implementations of RISC-V processors with different instruction sets, extensions and levels of optimizations. In another question **H. Boukabache** asked how time violations can be found with formal verification. **K. Ceesay-Seitz** responded that counter changes are evaluated in real time, which allows clock cycle counting and thus tracking of taint flow by tracking its propagation in clock cycles. **X. Fink** asked if the verification is done at runtime with outputs from the real system. **K. Ceesay-Seitz** answered that it is completely static, and the outputs of the system are modeled via the taint logic. **B. Fernandez Adiego** added that it is done at design time and not at runtime, which **K. Ceesay-Seitz** confirmed, adding that in case of a CPU all possible programs are evaluated during verification.

A. Pulli asked how the assumption that the software must guarantee constant time execution is implemented. **K. Ceesay-Seitz** responded that software programs are often verified to be constant time executing. Depending on the implementation the actual execution might be constant in time or not if executed on hardware. Both Intel and ARM give out instructions on how to ensure this. In the presented case, the classification will be made based on the software verification by exactly classifying which parameter can change the program counter and which not. **A. Pulli** noted that the instruction classification depends on the program itself. **K. Ceesay-Seitz** explained that the instruction classification is purely a property of the program and the goal is to detect which instruction was reading a parameter and which was changing the program counter. In another question **A. Pulli** asked how the presented approach could be scaled to more complex pipelines, e.g., a branch predictor or FinFET units. **K. Ceesay-Seitz** replied that the method is conceptually applicable to more complex pipelines. However, specific aspects such as auto-forward and taint injection would have to be specified differently. Especially for larger CPUs the challenge lies in specifying properties in a scalable way to avoid verification time explosion, which is an open problem many researchers are working on.

I. Lopez-Miguel asked about the type of formal proof that is used. **K. Ceesay-Seitz** responded that properties are specified as SystemVerilog properties for which the taint logic then does the specification. Then, the assumptions for the input and formal properties need to be written to allow verification in the model checker, which is done similarly to normal functional verifiers. **X. Fink** asked about the verification of specific instructions and the bottleneck between different CPU types. **K. Ceesay-Seitz** replied that the memory is completely abstract and valid for infinite program types. **H. Boukabache** asked whether formal verification for security can be generalized and used for other applications. **K. Ceesay-Seitz** agreed that it is applicable for all use cases that use information dependencies. **L. Felsberger** added that even hardware degradation leading to violations of information handling could be a possible use case.

2 PLCverif Extension: Verifying LTL Properties via Monitor Generation - X. Fink

X. Fink presented an extension of PLCverif, a tool developed at CERN for software verification of PLCs. The new extension allows the verification of linear temporal logic (LTL) properties, which is currently not supported by all verification backends. The presented approach extends the PLCverif toolbox with an algorithm for monitor-based verification of LTL properties using Bounded Model Checking (BMC) assertion-verification. Validation of the proposed algorithm on two CERN use cases demonstrated improvements of the verification times by up to one order of magnitude and additionally allows verification of previously unverifiable properties.

Discussion after the presentation:

L. Felsberger pointed out that NuSMV already supported LTL problems and asked about the benefit of adding support for other verification backends such as CBMC using PLCverif. **X. Fink** responded that this has two reasons: NuSMV supports a smaller set of language features, which can lead to inefficient models, and CBMC/ESBMC have increased performance on many verification instances. However, the presented approach is a lot faster due to better model reductions. **L. Felsberger** concluded that the proposed approach extends the range of applications and reduces the computation time. **K. Ceesay-Seitz** asked whether an adaption of the generated model to the model checking algorithms has been tried in PLCverif. **X. Fink** replied that this has not yet been studied, but could further enhance verification performance. **B. Fernandez Adiego** added that the models were changed slightly in the past, as some model checkers showed superior performance.

L. Felsberger thanked both speakers for their contribution. The next RAWG meeting will be on 23.01.2025 on the topic of reliable programming and verification. **L. Felsberger** closed the meeting wishing all participants a pleasant end-of-year break.