TrustFull: Trustworthy Fullstack Computing

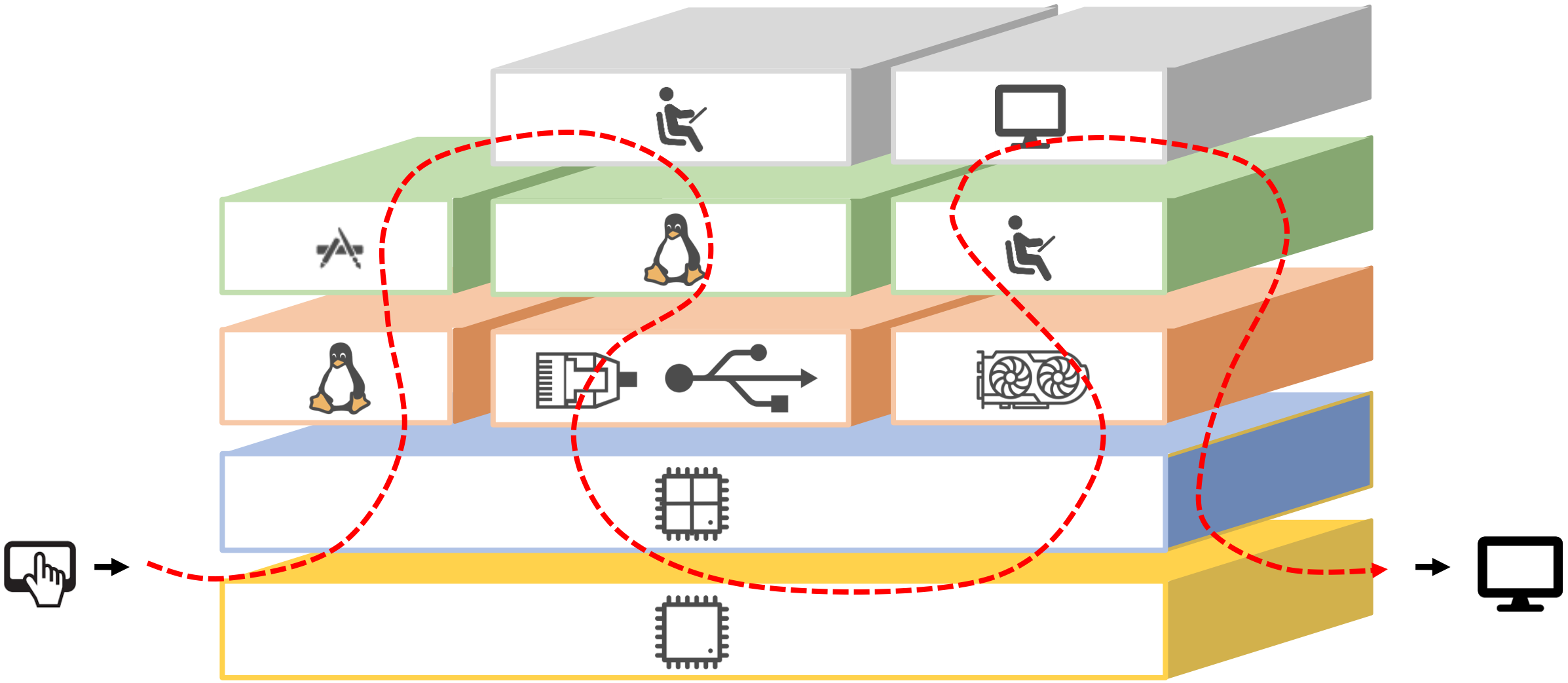
Mads Dam

Benoit Baudry, Roberto Guanciale, Martin Monperrus, Musard Balliu, Douglas Wikström, Karl Palmskog

Postdocs, PhD students, MSc students

KTH/EECS/TCS

SSF-Vinnova samverkanskonferens för cybersäkerhet 2021, Dec. 2021



Secure Fullstack Computing – The Challenges

#1: Size, complexity, legacy

- All three are fundamental enemies of security
- Separation of concerns
- Scalability where needed
- Precision where needed

#2: Lack of good models

- Current HW models are fundamentally flawed
- Do not capture the right properties
- Do not reflect the behaviour of real HW
- Example: Spectre

#3: Proprietary HW and SW

- Incomplete and inconsistent documentation
- Work with open source SW and HW as far as possible
- Example: RISC-V
- Or else models must be learned

#4: Lack of effective tools and methods

- Precision: To support critical security properties at OS/HW level
- Scalability: For automatic analysis, processing, and repair of large SW ensembles

The Project

WP1: Analysis and modelling

- Modelling and formal verification of low-level SW and HW

WP2: Application protection mechanisms

- Randomization and debloating, application security

WP3: Fault containment and recovery mechanisms

- Automatic repair, kernel and device security

WP4: Demonstrator

- E-voting, based on Open Verificatum

WP5: Exploitation and dissemination

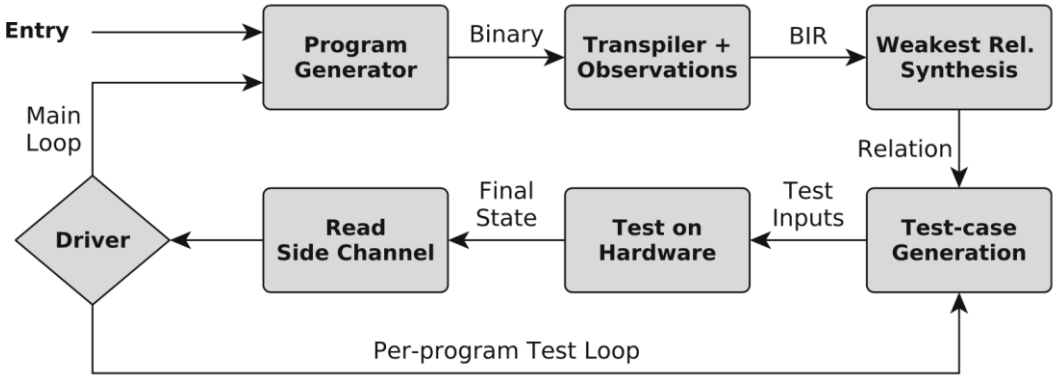
- Publications, open-source releases, outreach, education

Thank you!

Extra Slides

WP1 - Some Highlights

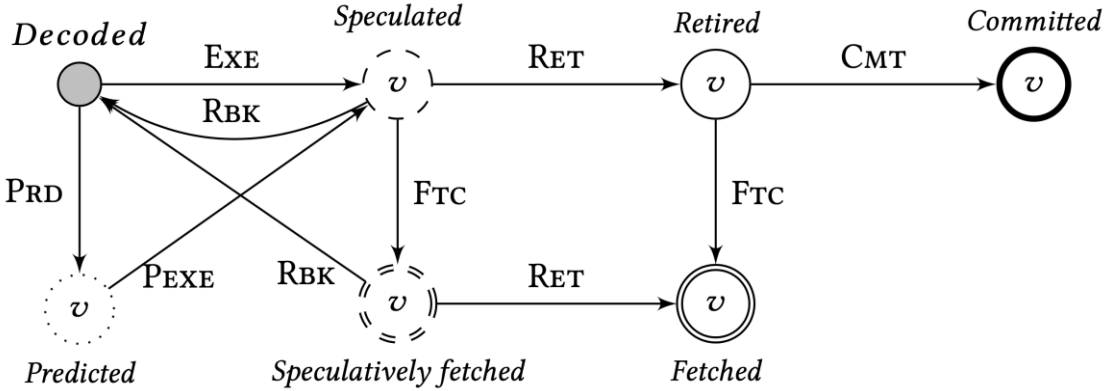
SCAM-V: Testing models against HW



- Formal models must reflect the behavior of the actual hardware
- Many problems under the hood - Spectre
- SCAM-V: New testing approach to validate models against real hardware

Modelling Side channels:

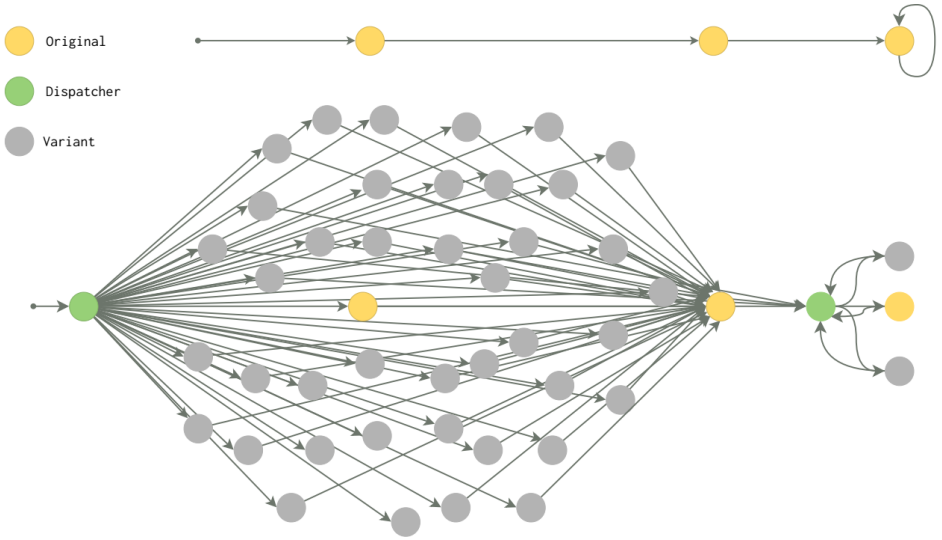
- Novel microinstruction-level modelling framework MIL
- Captures all known Spectre variants and then some
- New vulnerabilities identified
- Verified countermeasures



WP2 – Some Highlights

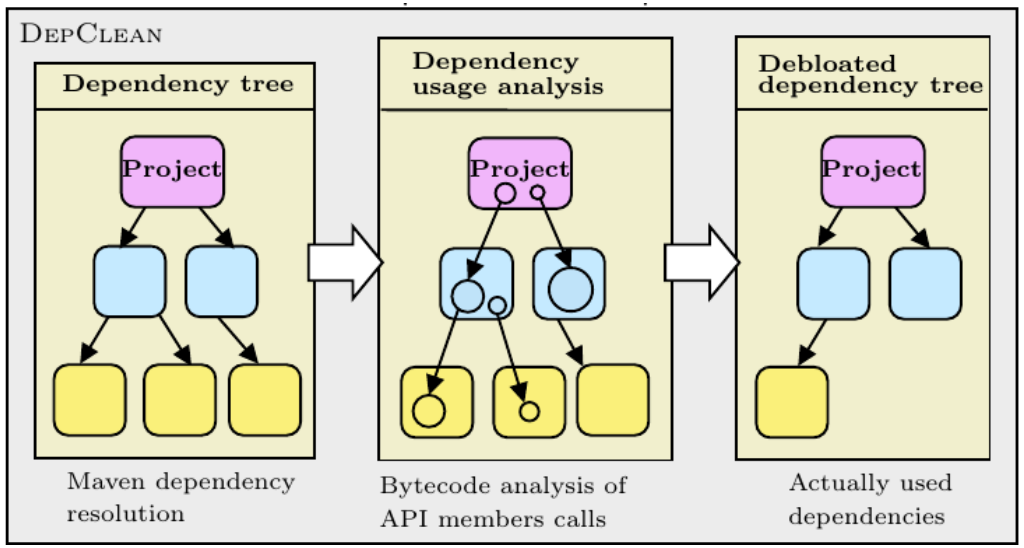
Automatic randomization:

- Automatic synthesis of Webassembly program variants
- Randomize crypto libraries on the world-wide edge platform of Fastly



Automatic code debloat

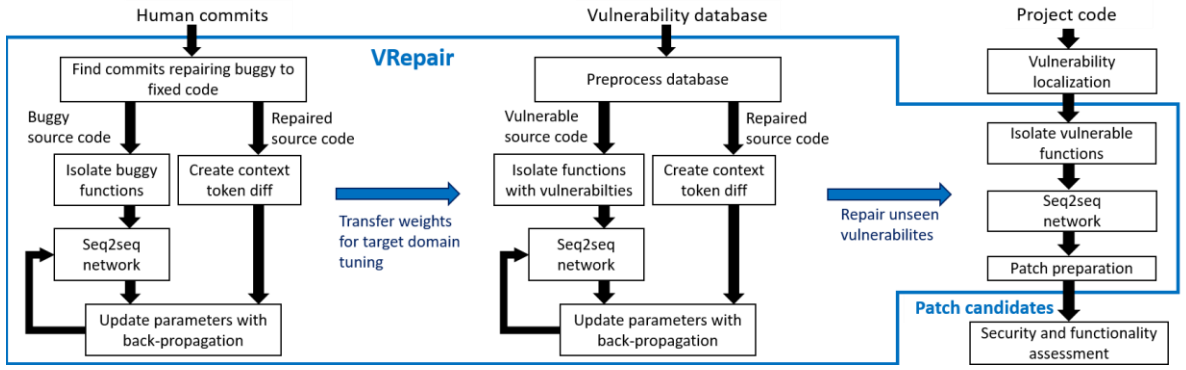
- Remove unnecessary software dependencies
- Static and dynamic code analysis on Java bytecode



WP3 – Some Highlights

Automatic repair of vulnerabilities

- Use neural networks to generate patches
- Collaboration with Colorado State University



Automatic repair of static warnings

- Focus on industry standard lightweight static analyzer SonarQube
- Uses metaprogramming to generate patches
- Repo at https://github.com/SpoonLabs/sorald/

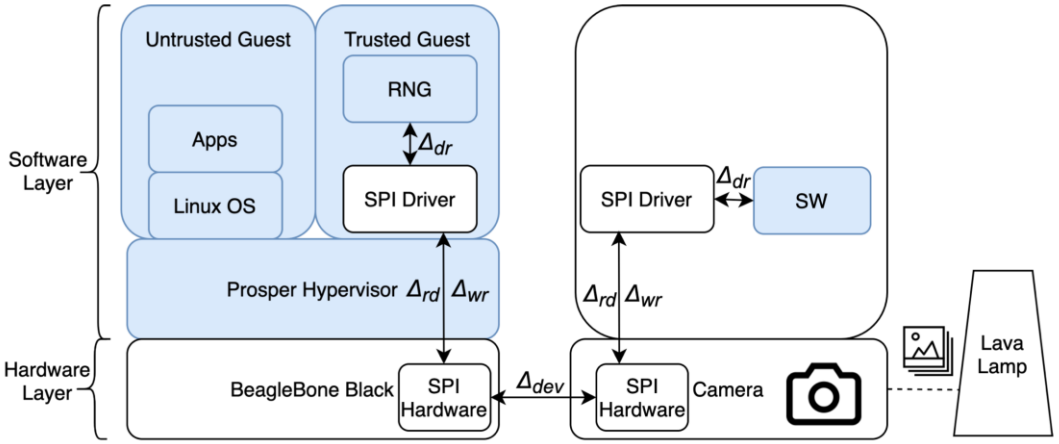
WP4 – Some Highlights

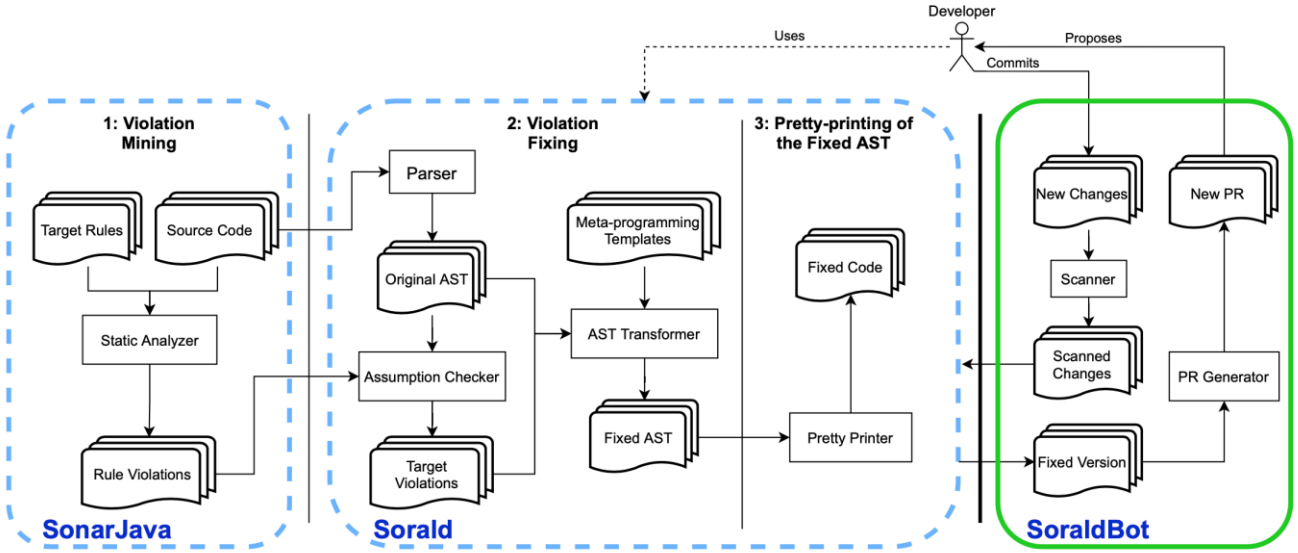
Demonstrator:

- Open Verificatum

Hardened random number generation

- Verified end2end sensor-to-application data flow for SPI devices





Automatic repair:

- Automated tool for patch suggestion in SonarQube
- Successfully applied to Verificatum