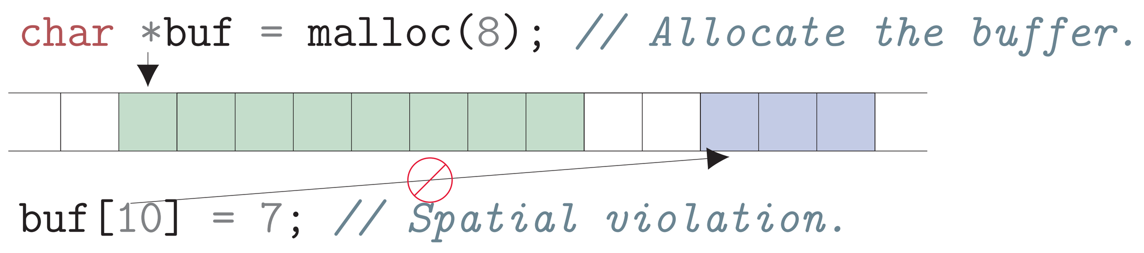
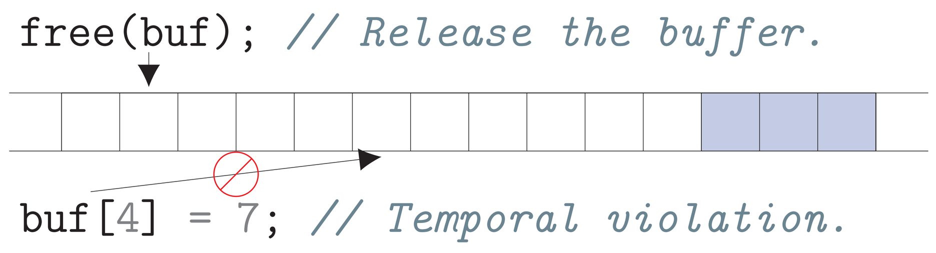
Securing Software From Hardware
A Survey on Thwarting Memory Corruption in RISC-V

Motivation

One of the ways to **hijack** a software program is by means of **memory corruption**. We distinguish between **two** types below:





To mitigate, **software**-based approaches exist, but they suffer from **high performance overheads** and **binary incompatibility** issues.

RISC-V

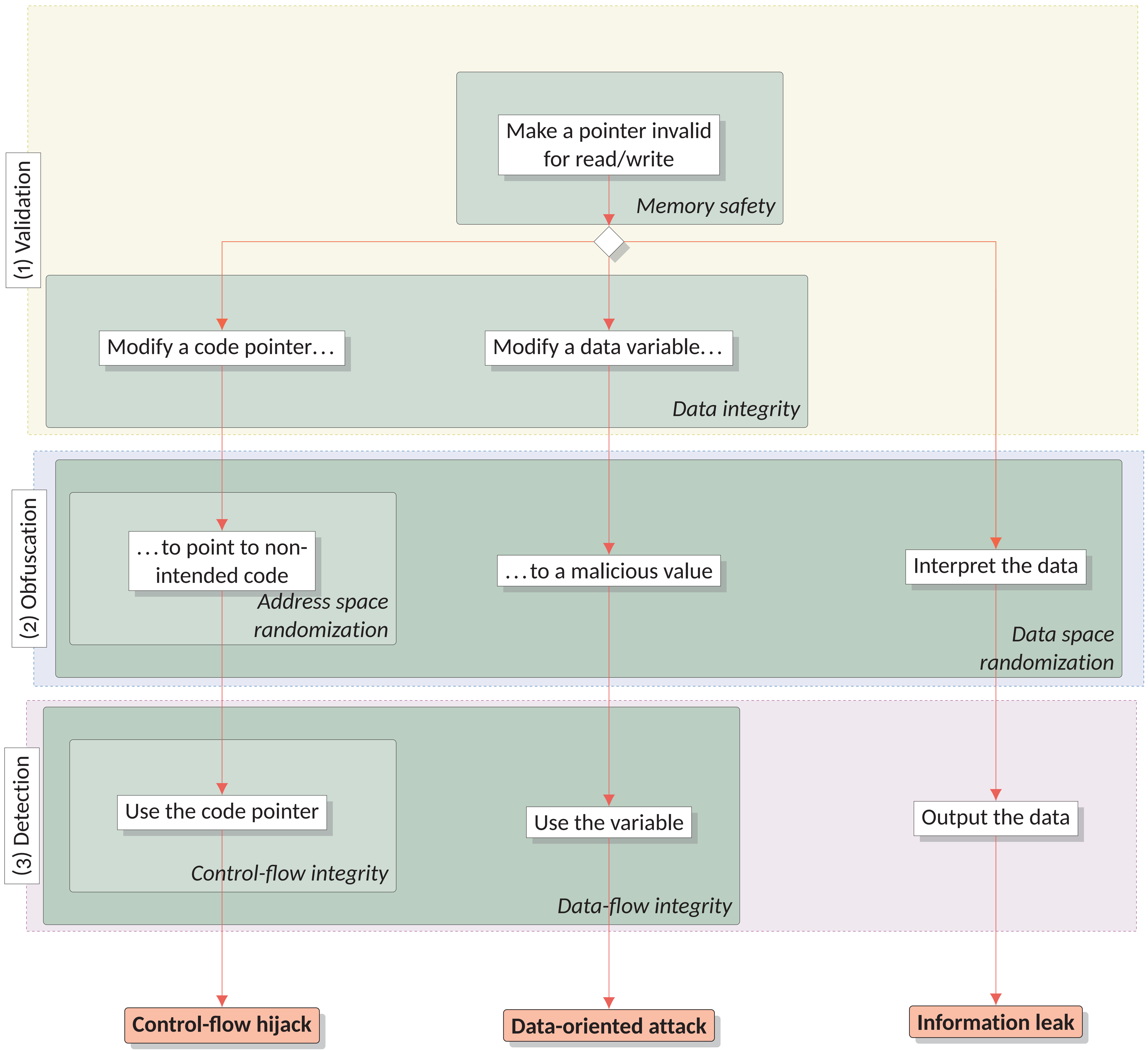
A novel and open **Instruction Set Architecture (ISA)** that allows you to:

1. Grab an **open hardware design** of a compatible processor.
2. **Customize it** with your new architectural feature.
3. Evaluate the performance, e.g. while **running Linux on FPGA**.

Research Question

Is a **hardware-assisted approach** to **mitigate memory corruption in software** worthwile?

Attack Paths & Protection Layers



Layer 1: Access Validation

Before a memory access is made, verify that:

1. the pointer is **supposed to access** this address.
2. the area has **not been freed** earlier.

These implementations focus on:

- **Storing** which pointers are **allowed** to access which memory.
- Placing **guards** around allocated buffers.

Layer 2: Data Obfuscation

Randomize the representation of code and/or data **pointers** and its **contents**.

Layer 3: Corruption Detection

Detect **upon reading** a specific value in memory whether it has been **corrupted previously**. Includes data shadowing, control-flow and data-flow graphs, crypto-based measures, runtime attestation and taint analysis.

Take-Home Messages

1. Data suggests that **a hardware-assisted approach has the potential** to improve on runtime overhead while keeping the increase in circuit area to a minimum.
2. Full protection is only possible with **compile-time information**, which does not solve the issue of binary incompatibility.
3. **Harmonization of evaluation** standards would be preferred, or at the very least **releasing the source code** to make it easier to adapt for research purposes.
4. Protection can still be **more fine-grained**, e.g. when overflowing into other fields of a `struct`.
5. Interesting to see **integration** in Trusted Execution Environments (TEEs) and whether the attack information can be used to **automatically craft security patches**.

Read more in ACM CSUR

University of Amsterdam
Informatics Institute
Complex Cyber Infrastructure
S3D

Marco Brohet (m.j.a.brohet@uva.nl), Francesco Regazzoni

CPS Summer School, 16–20 September 2024, Alghero (Italy)