APPCONTROL: ENFORCING APPLICATION BEHAVIOUR THROUGH TYPE-BASED CONSTRAINTS

Wim Vanderbauwhede[1], José Cano[1], Jan de Muijnck-Hughes[1], Cristian Urlea[1], Nobuko Yoshida[2], Adam Barwell[2], Klaus McDonald-Maier[3], Xiaojun Zhai[3], Sangeet Saha[3]

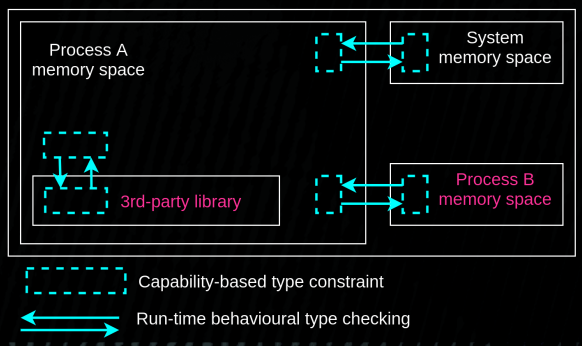
[1]University of Glasgow, [2]Imperial College London, [3]University of Essex

BACKGROUND

State-of-the-art techniques can be used to limit access privileges of third-party applications on certain computer systems. CHERI Capabilities provide fine-grained memory protection and isolation that scale better that competing techniques.

KEY IDEAS

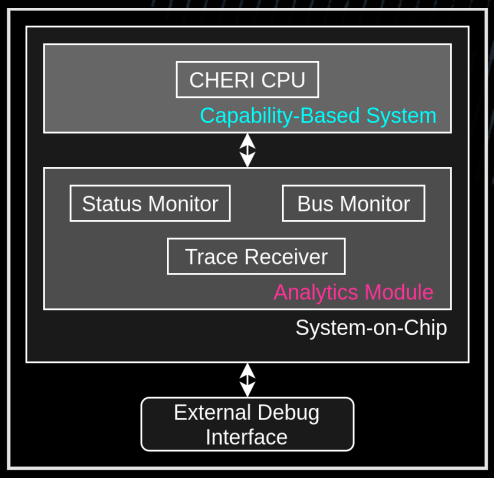
To secure program interaction we need to go beyond access privileges and ensure that a program **follows** the intended behavioural specification. Because Capabilities say nothing about **program behaviour**, we will also use **Behavioural Types** to capture the physical and behavioural structure of application interfaces.



- Behavioural typing supports **compile-time** checking of program behaviour when its implementation is known, and **run-time** checking of program behaviour when it is not known.
- We will leverage CHERI's Capabilities to ensure that behavioural types are not modified by parties unknown.

DEBUGGING INFRASTRUCTURE

- Design-by-specification will ensure correctness of behaviour, provided that the specification is correct. Debugging a specification-based system, demands the ability to debug the specification at run-time.
- Debugging system will include continuous diagnostics tools that allow to evaluate the system operation continuously and can identify hardware failure or unusual system behaviour.



- The on-chip debugger will monitor system busses, memory, CPUs, etc. Captured data will be used to extract useful features for analysis.

OBJECTIVES

- Develop enforceable specifications based on Capability Hardware
- Demonstrate the effectiveness of Enforceable Specifications based on Capability Hardware

SESSION TYPES FOR ENSURING CORRECTNESS OF COMMUNICATIONS

- Multiparty Session Types (MPSTs), traditionally a theory for specifying good communications in distributed and concurrent systems, give a global view (via a Global Type) of the behaviour across individual components within a system.
- Global Types are projected to Local Types, which provide a specification for the communications pertinent to a specific component within the system.
- MPSTs can be used to ensure communications between components conform to a desired specification.
- Rogue components trying to communicate in ways not permitted by the specification can be blocked from doing so via both static checks and dynamic monitoring enabled by MPSTs.
- We will combine and extend existing MPST theory and tooling to integrate with the behavioural type systems used to constrain general program behaviour, and with constraints expressible by CHERI Capabilities.

$$G = \mathbf{A} \to \mathbf{B} : \mathit{Count}(\mathit{count} : \mathtt{int}\{\mathit{count} \geq 0\}).$$
$$\mu \mathbf{t}(\mathit{curr} : \mathtt{int}\{\mathit{curr} \geq 0 \wedge \mathit{curr} \leq \mathit{count}\})\langle \mathit{curr} := 0\rangle.$$
$$\mathbf{B} \to \mathbf{C} \begin{cases} \mathit{Hello}(\mathit{it} : \mathtt{int}\{\mathit{curr} < \mathit{count} \wedge \mathit{it} = \mathit{count}\}). \\ \quad \mathbf{t}\langle \mathit{curr} := \mathit{curr} + 1\rangle \\ \mathit{Finish}(\mathit{it} : \mathtt{int}\{\mathit{curr} = \mathit{count} \wedge \mathit{it} = \mathit{count}\}).\mathtt{end} \end{cases}$$

WORK PACKAGES

- **WP1** Design a Type System starting from MPST and behavioural types.
- **WP2** Developer a compiler and run-time system that can be used to enforce behaviour on CHERI system.
- **WP3** Develop the required Operating System integration.
- **WP4** Develop the debugging system.
- **WP5** Demonstrate the effectiveness of our approach.

PROJECT WEB SITE

`https://dsbd-appcontrol.github.io/`

ACKNOWLEDGEMENTS

The authors thank the UKRI Digital Security by Design (DSbD) Programme for funding the AppControl project through grant EP/V000462/1.