UNIVERSITY OF CAMBRIDGE

Beyond SafeBet

CHERI-RISC-V & Transient Execution Attacks

Jonathan Woodruff, **Franz A. Fuchs**, Simon W. Moore, Robert N. M. Watson

RISE Annual Conference, London — 2nd December 2022

CHERI – Unforgeable References in Hardware

The CHERI capability: a hardware-defined & protected bounded pointer

Capabilities both **describe** & **grant access** to memory

Capabilities provide **memory safety**, and enable **compartmentalisation**

Over the last 10 years, CHERI has moved from academic research to commercial research and is on several paths to adoption

- ARM's Morello, funded by the UK Industrial Strategy Challenge Fund
- CHERI-RISC-V, funded by US Defense Advanced Research Projects Agency

  Picked up by Microsoft and Google research

What's the smallest variety of CHERI?

Security Research & Defense / By Saar Amar / September 6, 2022

The Portmeirion project is a collaboration between Microsoft Research Cambridge, Microsoft Security Response Center, and Azure Silicon Engineering & Solutions. Over the past year, we

CHERI & Transient Execution Safety (1)

Traditional CPU architecture is insulated from transient execution attacks

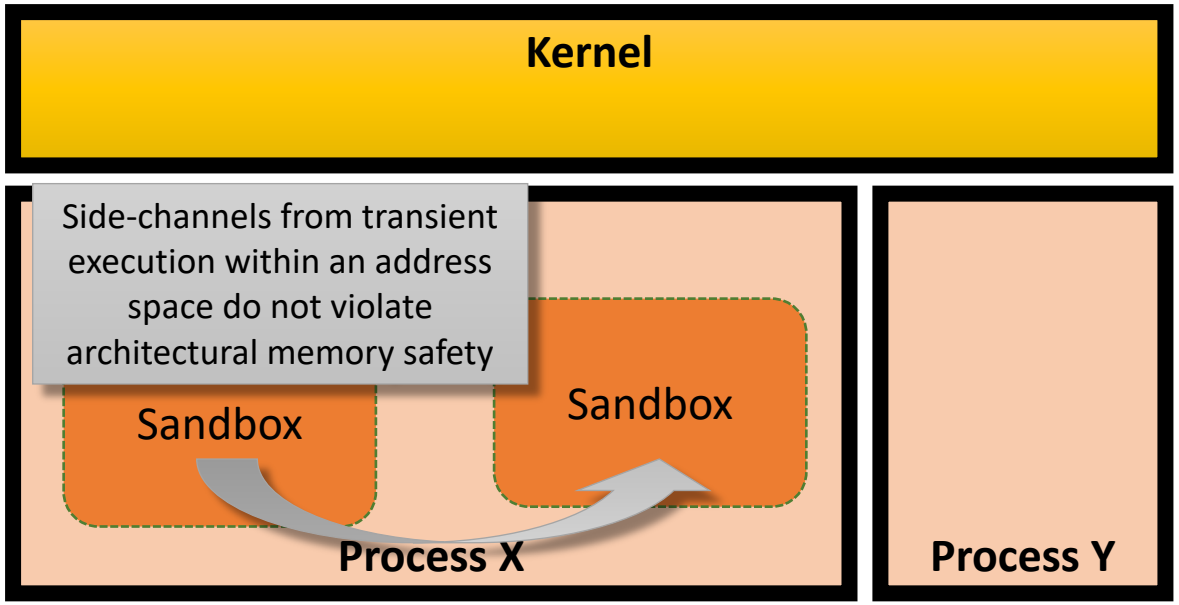
- Instruction Set Architectures (ISAs) notion of memory safety is limited to:
  1. Protection rings
  2. Virtual address spaces
- While some research proposed to block all side-channels from transient execution, mitigation mechanisms have been focused on attacks that compromise this architectural safety
  - Meltdown is fixed in hardware; BTB sharing between kernel and userspace is being addressed
  - On the other hand, Spectre attacks between user-space components has no hardware solution

CHERI provides ISA-guaranteed fine-grained memory safety

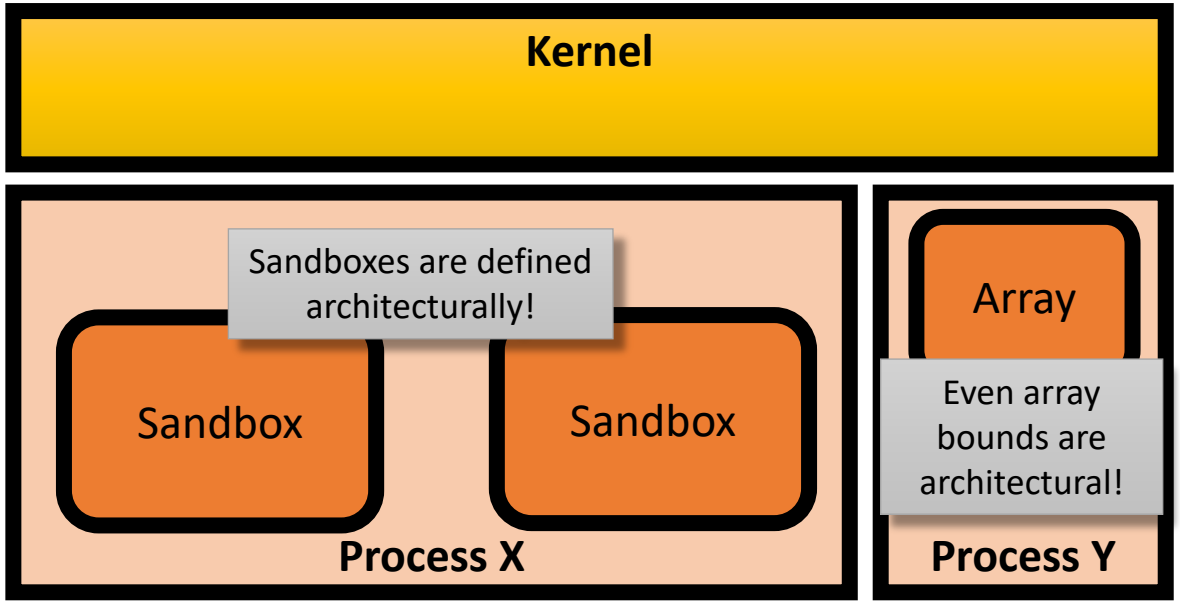
- With CHERI, out-of-bounds access is now an architectural threat
- With CHERI, software-defined compartment escape is now an architectural threat

CHERI & Transient Execution Safety (2)

Traditional Architectures (ISAs)



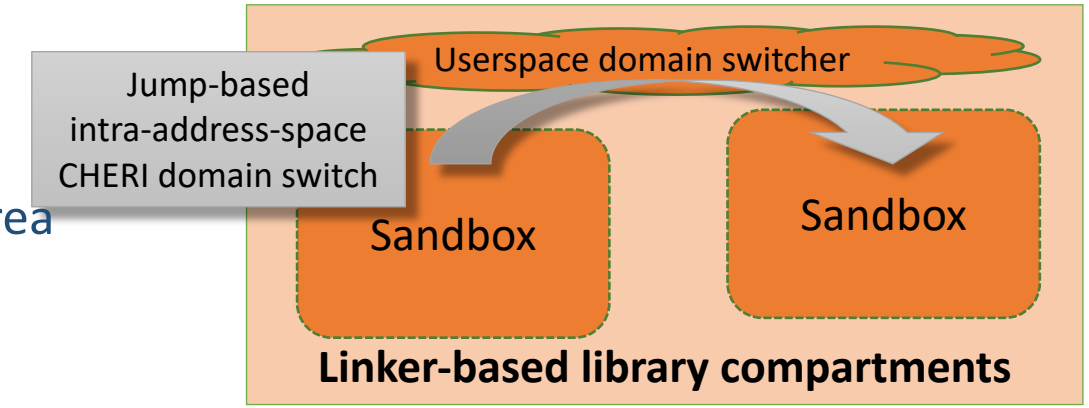
CHERI Architecture

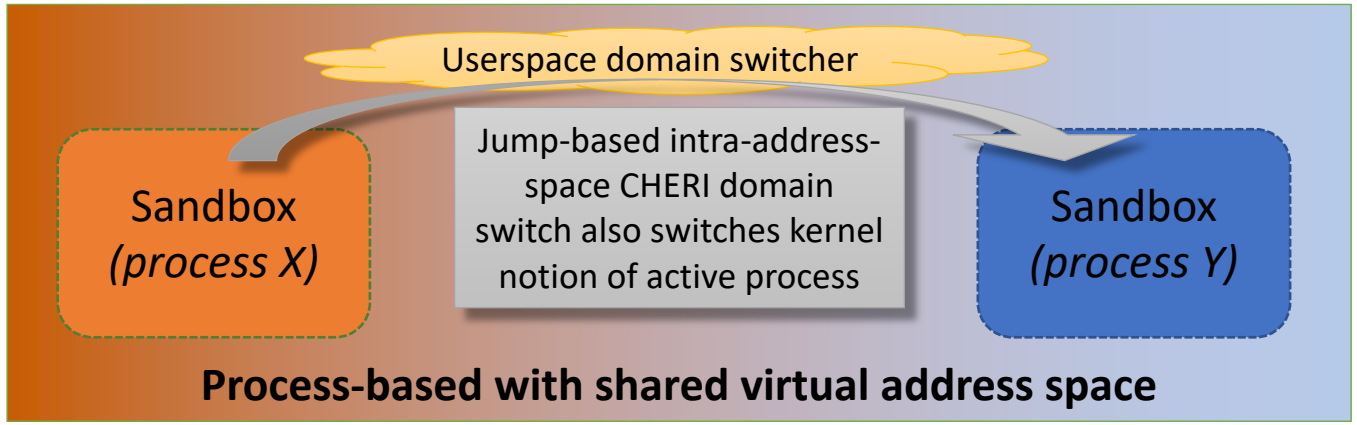


CHERI Compartmentalisation

CHERI compartmentilisation frameworks are a rich, developing area

- Several styles of CHERI compartmentalisation
  - Process-based, sharing an address space
  - Linker-based, lightweight library compartmentalisation
  - Frameworks for restructuring a single program
- What are the transient execution isolation expectations for each?
- How can we enforce these efficiently?





SafeBet Project

Capability Speculation Contract (CSC)

A CHERI processor must only load, store, or execute memory that is authorized by:

1. Capabilities in the committed register file.
2. Capabilities loadable from memory authorized by the set in Item 1, as well as capabilities transitively loadable through these.

Branching Speculation Contract (BSC)

1. The processor will not transiently execute instructions between two non-branching instructions; basic blocks will execute as-if sequentially in speculation.[a]
2. Direct jumps (unconditional with immediate target) will speculatively execute only the jump target, so that basic blocks can span direct jumps.
3. Direct branches (conditional with immediate target) will speculatively execute one of the two possible paths, with no other possibilities.
4. Instructions that cause exceptions will speculatively execute the non-faulting path.
5. Indirect branches (register target) will speculatively execute only previous architectural indirect-branch targets from the current compartment.[b]
6. Returns will speculatively execute the instruction after a previous call from the current compartment[c].

[a] Value prediction (e.g., memory disambiguation) can cause the same instructions to be replayed between non-branching instructions.
[b] Compartments are defined elsewhere in the architecture, by flushes on boundary, address space ID, or otherwise.
[c] Due to interrupts, lack of storage, and malformed programs, we cannot guarantee that a return will speculate to the correct return address.

- The SafeBet explored CHERI and transient execution vulnerabilities
  - Characterised transient execution vulnerabilities on the CHERI-Tooba core
  - Developed automated transient execution vulnerability discovery tooling
  - Developed architectural contracts to reason about program exposure to transient execution vulnerabilities
  - Developed contract enforcement mechanisms on CHERI-Tooba
  - Discovered CHERI-specific transient execution vulnerabilities and mitigations

BSC Jumps	BSC Value Prediction	Data-CSC	Inst-CSC
2.65%	1.65%	2.70%	0.35%

cbuildcap	*csetbounds*	*csetbounds exact*	*csetbounds immediate*	*cunseal*	*cinvoke*
0.10%	0.05%	0.20%	0.20%	0.20%	1.80%

Table 2: Percentage of sequences that produce counterexamples for each test generator; 2000 sequences for each. Data-CSC examples are broken down by CHERI instruction.

	inst-CSC	data-CSC	BSC Jumps	BSC Val. Pred.	TSC	BSC Exp.
ALU usage	99.85%	99.25%	99.57%	99.61%	99.90%	99.90%
FPU usage	87.05%	39.60%	0.00%	56.58%	98.15%	94.96%
MEM usage	0.00%	0.00%	0.00%	0.00%	0.00%	94.21%
Val. forward. usage	99.06%	96.60%	96.69%	97.97%	99.60%	99.60%
Excp front-end	0.00%	0.00%	0.00%	0.00%	0.00%	93.41%
Excp Rename	0.00%	0.00%	0.00%	0.00%	0.00%	94.81%
Excp ALU-0	69.74%	19.82%	0.00%	0.00%	0.00%	92.76%
Excp ALU-1	69.74%	19.87%	0.00%	0.00%	0.00%	92.76%
Excp MEM	53.42%	18.38%	0.00%	0.00%	91.56%	94.96%
Excp Commit	70.19%	20.20%	0.00%	0.00%	91.56%	94.96%
Redirect ALU	98.56%	18.47%	30.23%	0.00%	0.00%	0.00%
Redirect Commit	98.81%	38.95%	90.40%	79.01%	99.80%	99.80%
Dir Pred. usage	0.00%	0.00%	30.04%	0.00%	0.00%	0.00%
BTB Pred. usage	98.56%	17.91%	32.40%	49.18%	0.00%	0.00%
BTB Pred. Fail	85.27%	0.00%	9.32%	0.00%	0.00%	0.00%
RSB Pred. usage	85.47%	0.00%	0.00%	0.00%	0.00%	0.00%
RSB Pred. Fail	85.32%	0.00%	0.00%	0.00%	0.00%	0.00%
Mem Addr. Pred. Fail	0.00%	0.00%	0.00%	4.50%	0.00%	0.00%

Figure 6: Microarchitectural event coverage by generator.

UNIVERSITY OF CAMBRIDGE

SafeBet was funded by GCHQ under the RISE initiative (ref: 4213054)

CHERI – Increased Architectural State

- CHERI adds additional state to the architecture
  - Capabilities that lead to a transitive closure of permissions + memory
  - Sealing: Passing of capabilities and no random execution in the middle of a code block
  - Fine-grained compartmentalization
  - Knowledge of ”what is allowed and what is not”

- CHERI easily allows to reason about transient-execution
  - Come up with set of rules
  - Allows for (hardware) verification
  - Allows for transient execution attack mitigation mechanisms

Spectre v1 Mitigation in CHERI

- Spectre v1 attempts an out-of-bounds access
  - Speculative execution will execute the code in the if statement
  - Disabling the guard in the if condition
  - Allows arbitrary memory access in conventional systems

```
if(idx < size) {
  int a = arr0[idx];
  int b = arr1[a];
}
```

- CHERI can mitigate Spectre v1
  - Arrays are represented as capabilities
  - Out-of-bounds capability pointers are allowed, but dereferencing them is not
  - Will lead to a CHERI exception and the out-of-bounds access never happens
  - IMPORTANT: Microarchitectures must not lazily enforce that (see Meltdown)

CHERI – Increased Microarchitectural State

- Our vision for CHERI microarchitectures
  - More architectural state leads to more microarchitectural knowledge
  - Processors can make informed decisions on enhanced microarchitectural knowledge, which not only benefits security, but also performance

- Advantages in implementations
  - CHERI processors hold a subset of the capabilities in microarchitectural structures (register file, buffers,…) enabling to disallow forbidden (speculative) memory requests and related speculative decisions
  - Information about current compartment enables fine-grained microarchitectural security guarantees

CHERI – An Interface for Proveable Software Security

- ISA-level guarantees allow for:
  - Software to be assessed (or even verified) – possibly even automatically
  - Set of capabilities defines permissions
  - Combined with microarchitectural guarantees (see earlier slides), a CHERI system will be able to fully guarantee confidentiality

Conclusion & Questions

Jonathan.Woodruff@cl.cam.ac.uk

Franz.Fuchs@cl.cam.ac.uk