Review “CHERI”

Name: Wenhui Zhang

Paper Name：

Woodruff, J., Watson, R. N., Chisnall, D., Moore, S. W., Anderson, J., Davis, B., ... & Roe, M. (2014, June). The CHERI capability model: Revisiting RISC in an age of risk. In *ACM SIGARCH Computer Architecture News* (Vol. 42, No. 3, pp. 457-468). IEEE Press.

Contribution:

In this paper proposes a new RISC architecture CHERI (Capability Hardware Enhanced RISC Instructions), which implements a hybrid capability model with memory protection. CHERI uses 31 bits for permission in a 256bits memory, which provides fine grained protection within address spaces. A CHERI enabled FPGA, with FreeBSD executing on it is presented. Also an adapted CHERI version LLVM is released.

Motivation:

Programmers rely on application isolation techniques to mitigate inevitable vulnerabilities. They are used to use sandboxing to deploy applications with least privilege principle. However, as granularity increases, old solutions are too coarse to get exploit triggering packets filtered. Also utilization of sandboxing is slows down the runtime performance. IPC-linked processes introduces security issues as well. Thus there is call for hardware enhanced capability checking as a basis for secure software development, like sandboxing.

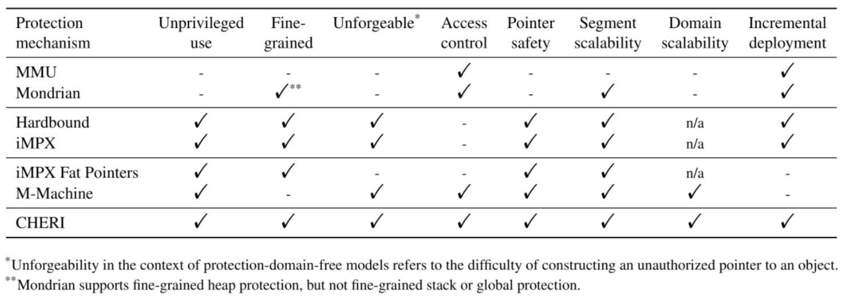
Related works:

Works like M-Machine, Mondrian, iMPX, Hardbound are proposed.

Methodology:

CHERI extends contemporary 64-bit RISC architectures with architectural capability. It utilizes capability registers (permission bits of 31) and tagged memory (base tag of 64bits, length tag of 64 bits) for a fine granularity of memory (256 bits). It protects pointee code/data through fine grained bounds and permissions. CHERI extends pointers with (1) tags to protect in memory capabilities by dereferencing untagged capability and throwing exceptions; (2) in memory overwrite clears capability tag; (3) bounds limit on pointers for range of accessible addresses; (4) operations(e.g., load, store, fetch) have to permitted; (5) encapsulation is immutable and non-dereferenceable.

Results:



Protection mechanism	Unprivileged use	Fine-grained	Unforgeable*	Access control	Pointer safety	Segment scalability	Domain scalability	Incremental deployment
MMU	-	-	-	✓	-	-	-	✓
Mondrian	-	✓**	-	✓	-	✓	-	✓
Hardbound	✓	✓	✓	-	✓	✓	n/a	✓
iMPX	✓	✓	✓	-	✓	✓	n/a	✓
iMPX Fat Pointers	✓	✓	-	-	✓	✓	n/a	-
M-Machine	✓	-	✓	✓	✓	✓	✓	-
CHERI	✓	✓	✓	✓	✓	✓	✓	✓

*Unforgeability in the context of protection-domain-free models refers to the difficulty of constructing an unauthorized pointer to an object.

**Mondrian supports fine-grained heap protection, but not fine-grained stack or global protection.

Table 2: Comparison of address-validity, pointer-validity (table-based), and pointer-validity (fat-pointer based) models

Take away:

Hardware security extension provides faster and more fine grained protection. Learned evaluation metrics for security oriented architecture designs.