CheriBSD
A memory safe POSIX OS


SRI International, University of Cambridge, Microsoft Research, Google, Inc

Approved for public release; distribution is unlimited. This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts FA8750-10-C-0237 ("CTSRD") and HR0011-18-C-0016 ("ECATS"). The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.
Introduction to CHERI

- CHERI introduces a new hardware type: the capability
  - In addition to integer and floating point
- CHERI capabilities grant access to bounded regions of virtual address space
  - Protected by tags in register and memory

Watson, et al. **CHERI: a research platform deconflating hardware virtualization and protection.** RESoLVE 2012.

Architectural CHERI capabilities

CHERI capabilities extend pointers with:

- **Tags** protect capabilities in registers and memory
- **Bounds** limit range of address space accessible via a pointer
- **Permissions** limit operations – e.g., load, store, instruction fetch

Actual implementation is compressed to **128-bits** with floating-point bounds
• All memory access via capabilities
  • Explicit (new instructions):
    • Capability load, store, branch, jump
  • Implicit (legacy ISA):
    • via Default Data Capability (DDC) or Program Counter Capability (PCC)
• Capabilities are used and manipulated in capability registers by capability instructions
  • Manipulations are monotonic (can only reduce bounds and permissions)
• Capabilities can be stored in memory, protected by tags
Capabilities as C pointers

- CHERI capabilities are designed for use as C pointers
  - Allowed to be out of bounds between dereferences
  - Can store 64-bit integers (untagged)
- Two compilation modes:
  - Hybrid: \texttt{__capability} annotation applied to select pointers
  - Pure-capability: all pointers are capabilities

CheriABI: Pure-capability process environment

• Built on CheriBSD (FreeBSD modified for CHERI)
• All pointers are capabilities
  • Including system call arguments and return values
• Bounds are minimized
  • C-language objects
  • Pointers provided by the kernel
• Goal: run pure-capability programs with simple recompilation


Abstract capabilities

How should the systems programmer think about bounds?

New concept: abstract capability
• Set of permissions of the process
• Tracks ghost state across swapping, etc
• Constructed and maintained by a collaboration of the kernel and language runtime
System startup

Power-on state

<table>
<thead>
<tr>
<th>Registers</th>
<th>DDC</th>
<th>PCC</th>
<th>C1-31</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWX</td>
<td>0x0</td>
<td>-</td>
<td>0xFF…FF</td>
</tr>
<tr>
<td>NULL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory

All tags clear

Early boot

<table>
<thead>
<tr>
<th>DDC</th>
<th>PCC</th>
<th>C1-31</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-W</td>
<td>0x0</td>
<td>0xFF…FF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UserRoot</th>
<th>SwapRoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWX</td>
<td>RWX</td>
</tr>
<tr>
<td>0x0-0x0000007F…FF</td>
<td>0x0 - 0xFF…FF</td>
</tr>
</tbody>
</table>
Execve

Initial register values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DDC</td>
<td>NULL</td>
</tr>
<tr>
<td>PCC</td>
<td>RWX</td>
</tr>
<tr>
<td>CSP</td>
<td>RW-</td>
</tr>
<tr>
<td>C03</td>
<td>RW-</td>
</tr>
</tbody>
</table>

UserRoot  RWX 0x0-0x000007F...FF

Kernel

Userspace

Process arguments

auxargs

environ

argv

Arg & environ strings

Thread Stack

Program binary

Run-time linker
Virtual-memory system

• Programmer visible:
  • Provides capabilities to newly mapped regions via `mmap()` and `shmat()`
  • Alters and frees mappings

• Abstract capability maintenance:
  • Ensures correct virtual to physical mappings
  • Preserves stored capabilities in swapped pages
Run-time linker

- Loads and links dynamic libraries
- Resolves symbols and synthesizes capabilities
- Jumps to program entry point
- Provides on-demand loading of libraries and supports exception handling
C runtime

- Objects allocated by `malloc()` are bounded to requested size
- `realloc()` adjusts bounds or allocates new storage as required
- Thread-local storage is bounded
  - Currently to per-thread storage
- Compiler generated code sets bounds on stack, automatic, and global objects
System calls

```
read(fd, buffer, nbyte);
```

copyout(kaddr, buffer, len);
...
kern_readv(td, fd, {buffer, nbyte});
sys_read(td, uap);

Kernel

Userspace

TCB

<table>
<thead>
<tr>
<th>v0</th>
<th>SYS_READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>a0</td>
<td>fd</td>
</tr>
<tr>
<td>c3</td>
<td>RW- buffer</td>
</tr>
<tr>
<td>a1</td>
<td>nbyte</td>
</tr>
</tbody>
</table>

Thread Stack

buffer
Required source code changes

• Userspace: 1% (~200) of files required changes
  • Concentrated in libraries
  • Most programs require no changes
• Kernel: <6% of files (~750) required changes
  • Pervasive changes to iovec, signal handlers, network interface ioctl handlers
  • A pure-capability kernel could reduce changes

• Many changes improve code quality
  • Upstreaming to FreeBSD and other projects often possible
Capability bounds minimization (OpenSSL)

Most capabilities bound small regions (<<1 page)

Small number of whole shared-object references remain in startup code

Stack references

Better

Number of capabilities

Size

0 2^2 2^5 2^8 2^11 2^14 2^17 2^20 2^23

0 20000 40000 60000 80000 100000 120000

all malloc glob relocs syscall kern
stack exec
Micro-benchmark performance generally acceptable

- <10% overhead in most cases
- Graph excludes crypto and bit-manipulation outliers
Conclusions

• Full UNIX-like operating system with spatial and referential memory safety
  • Covers programs, libraries, and linkers
  • Kernel access to user memory
• Some fundamental operating system changes required
  • Generally non-disruptive
• 3rd-party software works:
  PostgreSQL database, Webkit
Further Reading

http://cheri-cpu.org/


This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for public release. Distribution is unlimited.