

CTSRD

CRASH-WORTHY
TRUSTWORTHY
SYSTEMS
RESEARCH AND
DEVELOPMENT

CTSRD Project Briefing

Robert N. M. Watson (Cambridge)
Peter G. Neumann (SRI) Simon W. Moore (Cambridge)

DARPA CRASH PI Meeting
San Diego, California, USA
14 January 2014



Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.



UNIVERSITY OF
CAMBRIDGE

CTSRD at the PI Meeting



Dr Peter G.
Neumann



Dr Robert N. M.
Watson



Dr Simon W.
Moore



Dr Jonathan
Anderson



Dr David
Chisnall



Dr Nirav
Dave



Mr Brooks
Davis



Mr Rance
DeLong



Dr Khilan
Gudka



Dr Theo A.
Markettos



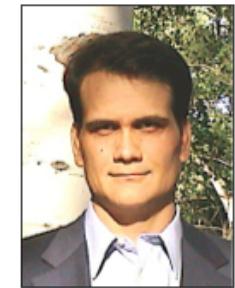
Mr Ed
Maste



Dr Michael
Roe



Mr Colin
Rothwell



Mr Stacey
Son

CTSRD

- Spans security, CPUs, OS, compilers, languages, program analysis/transformation, HW/SW formal methods.
- Clean-slate design violates some current conventions, in exchange for dramatic security improvements.
- Capability-based CPU protection and compartmentalization features mitigate known and unknown vulnerability classes.
- Hybrid model facilitates incremental SW adoption.
- Program analysis and transformation techniques improve software TCB correctness, utilize new CPU features.
- Formal methods link models, hardware, and software.



CTSRD project elements

- Capsicum, compartmentalization, and CTSRD
- Capability Hardware Enhanced RISC Instructions (CHERI)
 - CHERI ISA and hardware compartmentalization prototype
 - CHERI platform: tablet, CheriCloud, peripherals, etc.
 - CHERI software: CheriBSD, CHERI Clang/LLVM, apps
 - Architectural extraction, verification of Bluespec (Smten)
 - ISA-level proofs and automated test generation
- Security-Oriented Analysis of Application Programs (SOAAP)
- Temporally Enhanced Security Logic Assertions (TESLA)

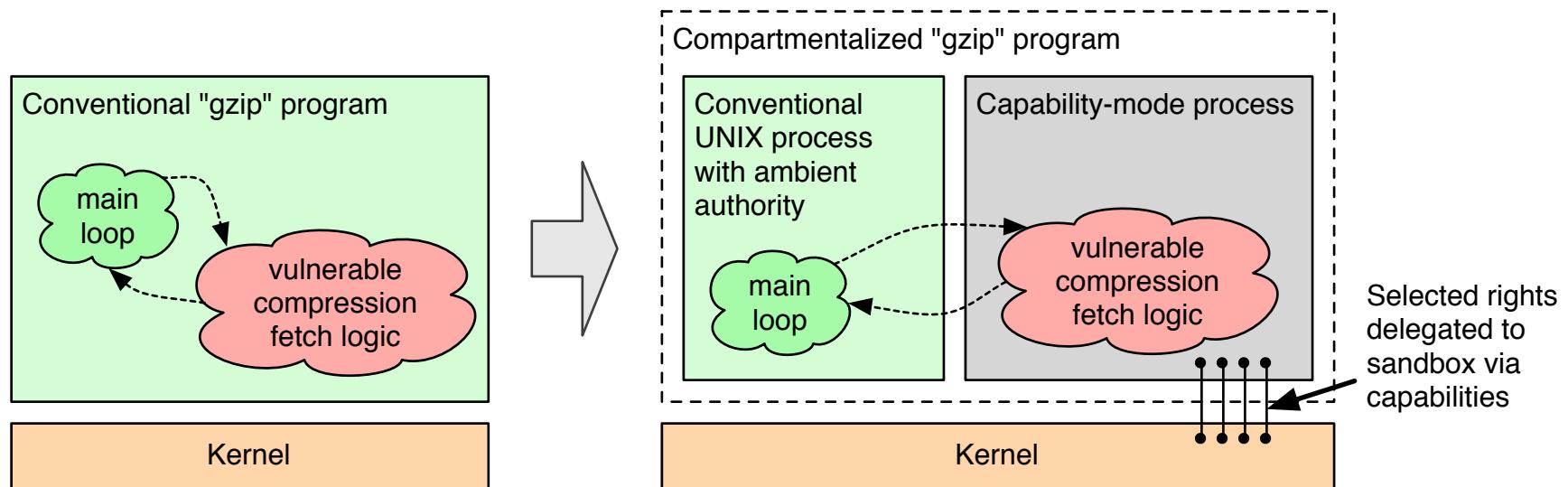
August 2013 CTSRD/MRC2 meetings, Cambridge, UK



COMPARTMENTALIZATION FOUNDATIONS

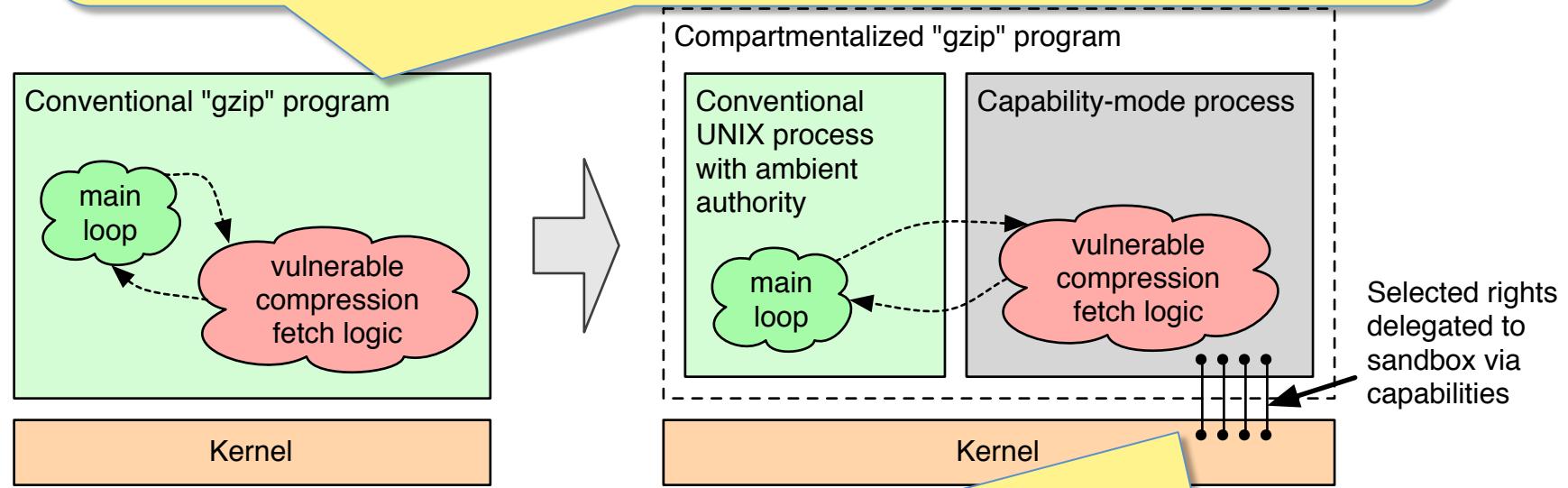


Application compartmentalization



- Compartmentalization decomposes software into isolated components.
- Each sandbox runs with only the rights required to perform its function.
- This model implements the principle of least privilege.

When a conventional application is compromised, ambient rights are leaked to the attacker, e.g., network and file system access.



Compromising a compartmentalized application yields only held rights to the attacker.

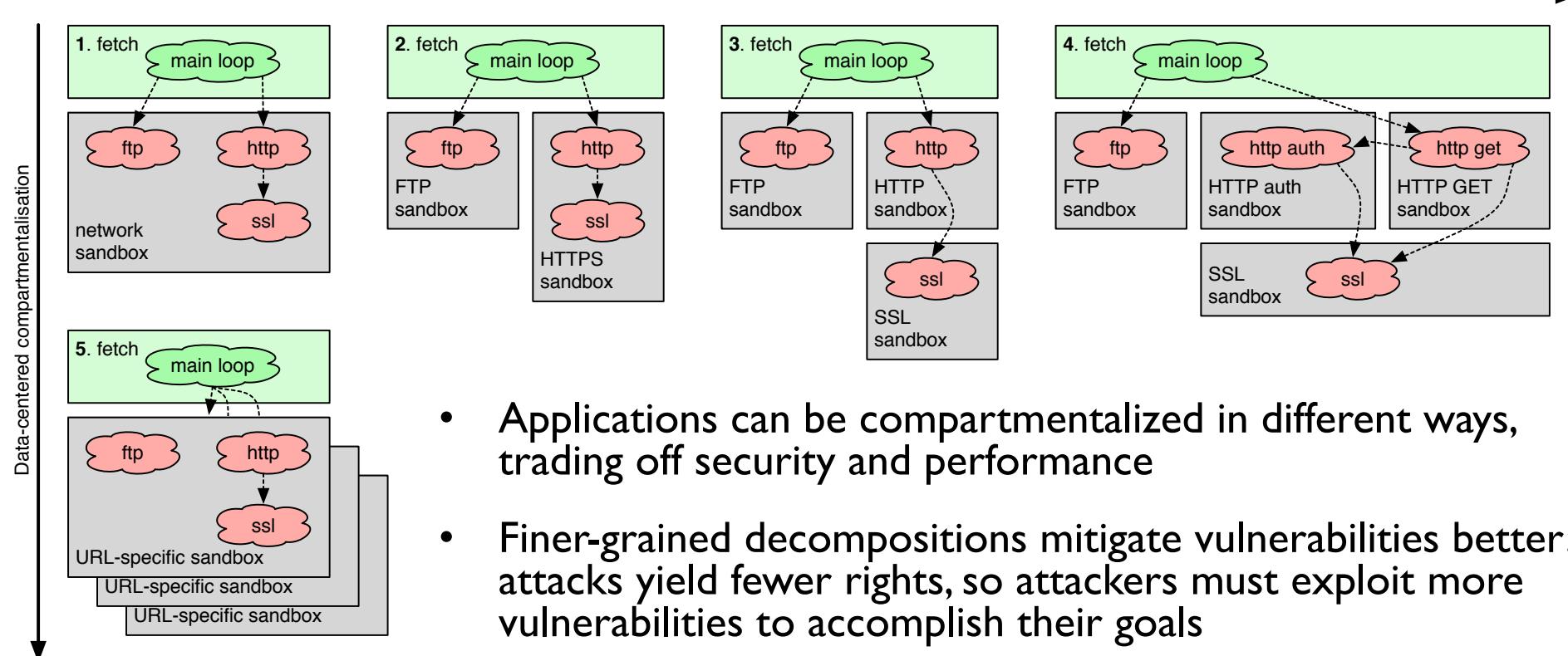
As vulnerabilities yield fewer rights, attackers must exploit many vulnerabilities to meet their goals.

Capsicum update



Capsicum

- Hybrid capability model: OS APIs for application compartmentalization
- Joint Cambridge/Google project
- Experimental feature in FreeBSD 9.x; out-of-the box in 10.0 (RSN)
- FreeBSD Foundation, Google
 - Funded projects will continue in 2014
 - Growing number of FreeBSD programs are using Capsicum out-of-the-box: tcpdump, auditdistd, hastd, etc.
 - Casper framework offers services to sandboxes (e.g., DNS, socket server)
- Google has published a Linux port prototype and hopes to upstream



- Applications can be compartmentalized in different ways, trading off security and performance
- Finer-grained decompositions mitigate vulnerabilities better: attacks yield fewer rights, so attackers must exploit more vulnerabilities to accomplish their goals
- Ideally, web browsers would use hundreds/thousands of sandboxes: one for each image, script, etc.
- However, CPUs support few simultaneous processes; e.g., Google Chrome reuses up to 20 sandboxes, one per tab
- As a consequence of CPU design, malware in a webmail image attachment can access a user's entire mailbox

Capability hardware enhanced RISC instructions

CHERI PROCESSOR

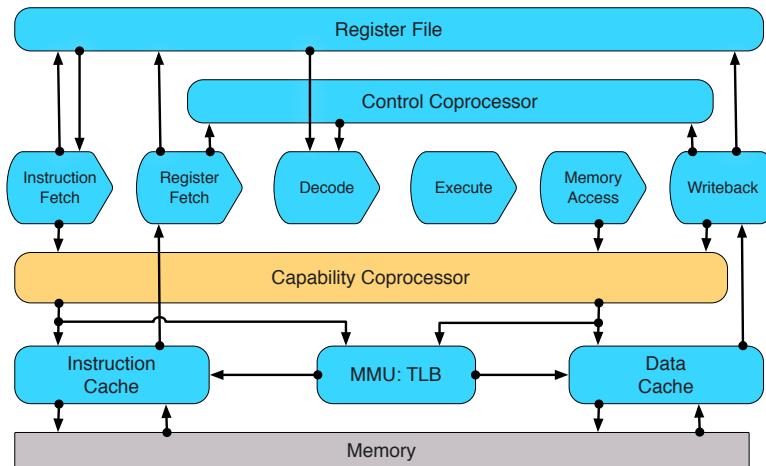


II

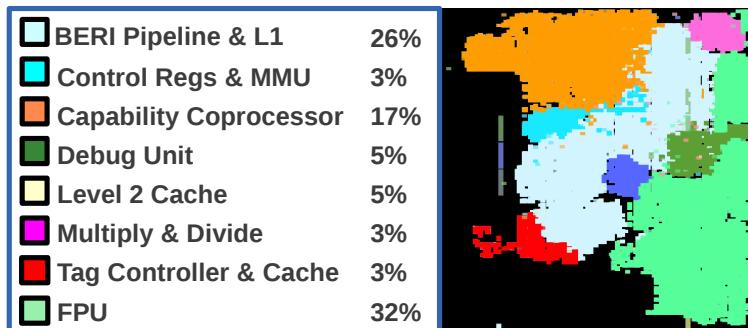


UNIVERSITY OF
CAMBRIDGE

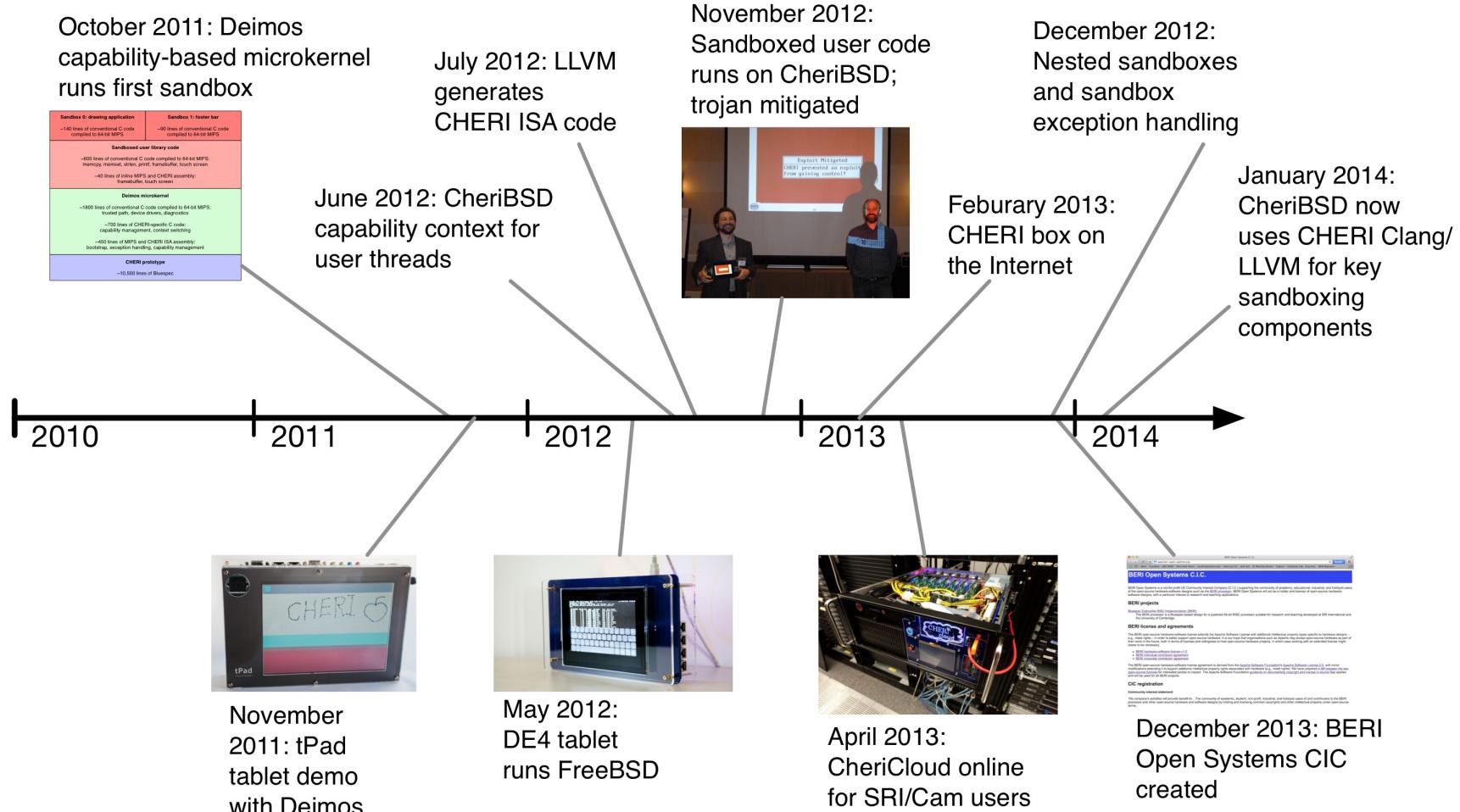
Capability hardware enhanced RISC instructions (CHERI)



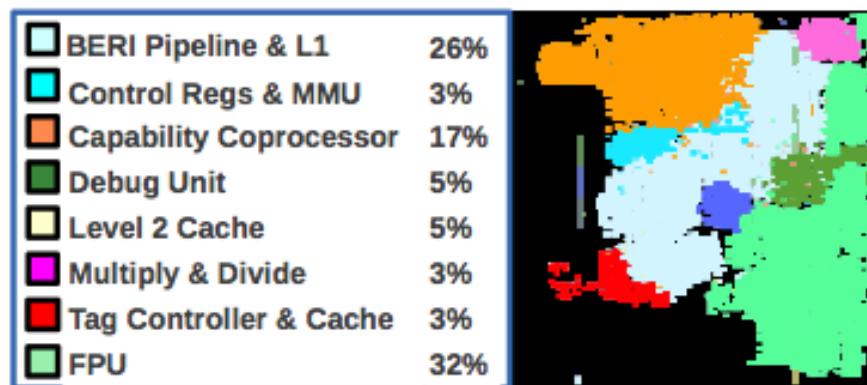
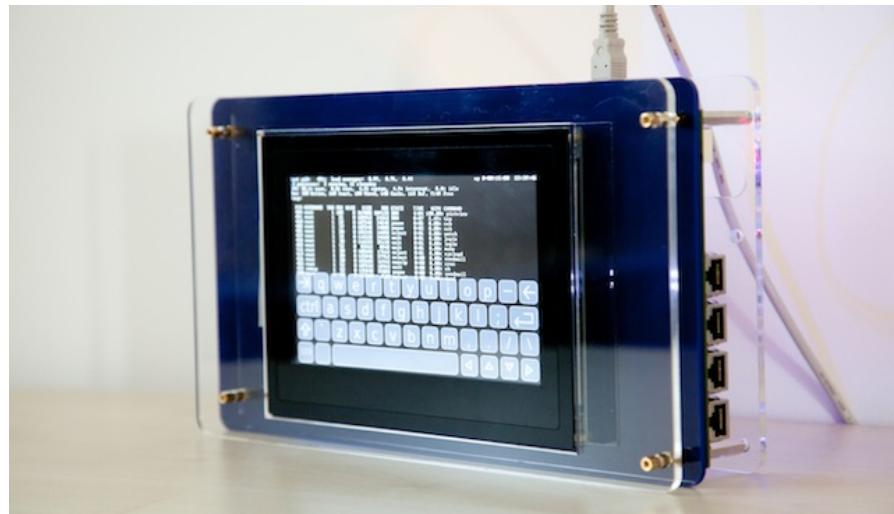
- CHERI hybrid capability model:
 - Fine-grained memory protection
 - In-address-space sandboxing
- Extends 64-bit MIPS ISA
- Haskell-derived Bluespec System Verilog HDL; synthesizes to Altera and Xilinx FPGAs
- Fully pipelined; multithreaded and multicore under development
- Extensive test suite and tools
- ISA and design subject to new formal analysis
- Shortly to be released as open source



CHERI development timeline



CHERI hardware platform



- CHERI prototypes
- Tablet prototype: CPU, DRAM, battery, flash, touchscreen, HDMI, Ethernet
- In-field CPU, OS updates
- CheriBSD OS, CHERI SDK
- CHERI demonstrations
 - E.g., fine-grained compartmentalization in CheriPoint presentation package

CheriCloud



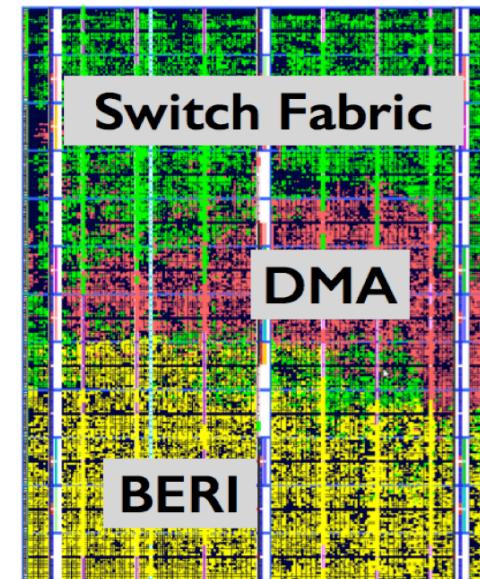
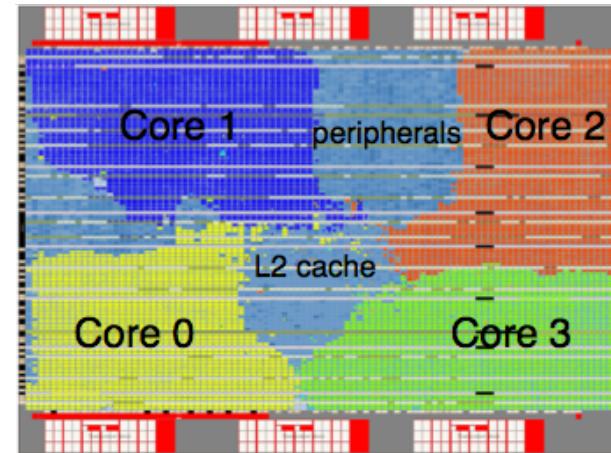
- Centralized facility supporting remote CHERI software development for CTSRD and MRC2 projects
- 7 x DE4 FPGA boards in 4U
- CHERI CPU + CheriBSD with Ethernet on each DE4
- Reset and serial console
- SSH into a live CHERI system; off-the-shelf open-source applications
- Total end-user transparency!

CHERI enhancements (CTSRD)

- CHERI ISAv2.I enhancements to object-capability invocation, software debugging features
- FPU – particularly useful for Olden benchmarks
- Improved hardware ISA-level tracing + CheriVis
- CHERI2 now fully implements ISAv2.I
- Annabella release in July 2013
- Multithreaded CHERI2 boots BSD in simulation
- Multicore CHERI in testing

CHERI enhancements ((MRC)²)

- DARPA MRC sister project also using and enhancing CHERI
- Multithreading and multicore
- Multi-FPGA interconnect
- AXI bus conversion
- NetFPGA 10G
- CPU Tracing enhancements
- FPU maturity
- BlueSwitch



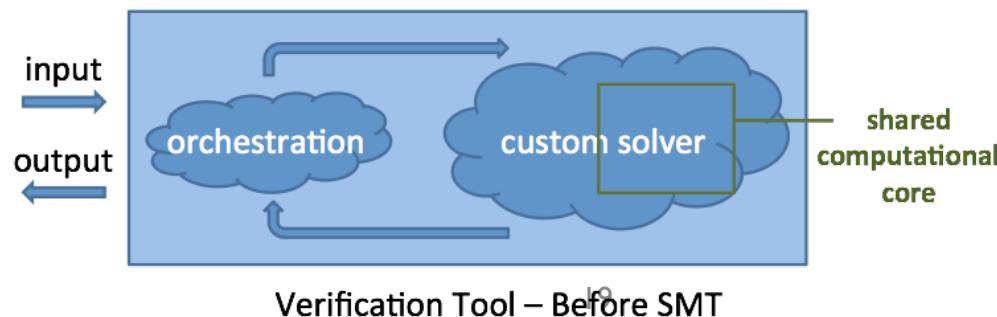
CHERI formal verification: ISA model

- Formal model of ISA described in SAL model checker
- Bluespec CHERI and CHERI2 capability units are automatically tested against the SAL model
- New: we can now automatically translate the SAL model into PVS
- Using PVS, we can prove “memory safety” for a CHERI ISA subset
- Future work: prove security properties of full model
- Future work: prove security properties of key software TCB elements (e.g., CCall, Creturn)



CHERI formal verification: Bluespec analysis / Smten

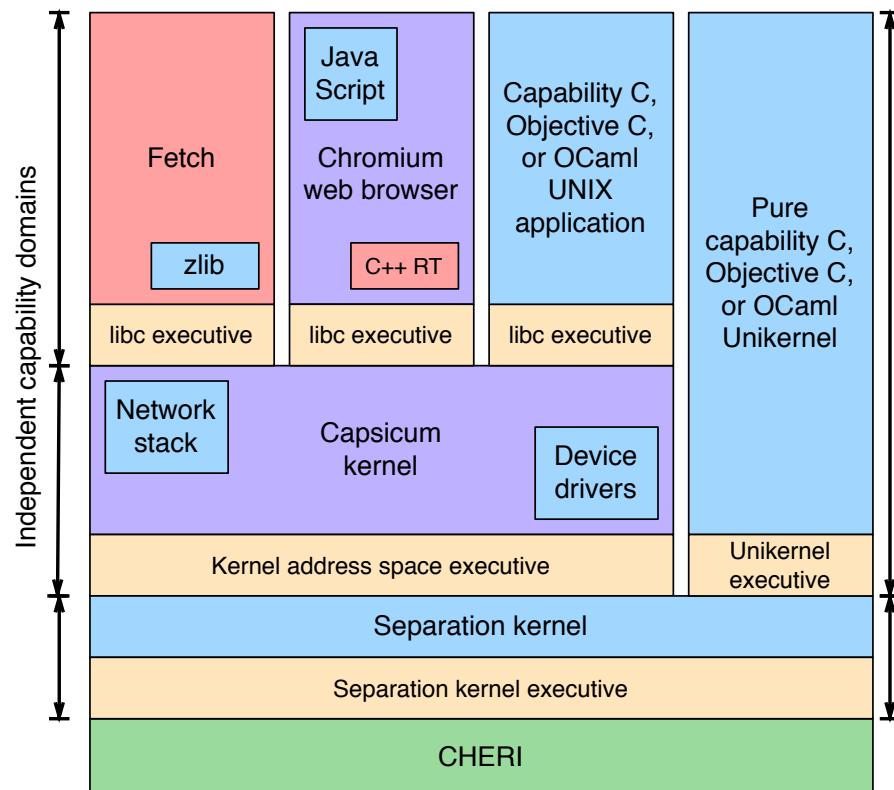
- Smten: Automatic Translation of High-level Symbolic Computations into SMT Queries
- Motivation: SMT solvers are widely used for model checking, automated theorem proving and test generation, but translating a model into an SMT form is tedious
- Solution: Smten is a high-level, purely functional language, with syntax and features borrowed heavily from Haskell which greatly helps translation of models into SMT queries
- Initial work published at CAV'13
- See Nirav Dave during the poster session for details.



CHERI SOFTWARE



CHERI software model



- Legacy application code compiled for general-purpose registers
- Hybrid code blending general-purpose registers and capabilities
- High-assurance "pure" capability code
- Per-address space memory management and capability executive

- Fine-grained userspace memory protection, in-process sandboxing
- MIPS/CHERI binaries tightly integrated (e.g., CHERI library in MIPS binary)
- Compiler allows pointers to be replaced with tagged, bounds-checked capabilities
- In-progress CHERI debugger
- OS support for model, tracing tools, etc.
- Userspace sandbox model, class libraries, components, monitoring tools
- CheriBSD on [github](#) so more easily accessible to downstream users
- BERI support + drivers shipping as FreeBSD 10.0 embedded target in weeks

CHERI extensions to FreeBSD

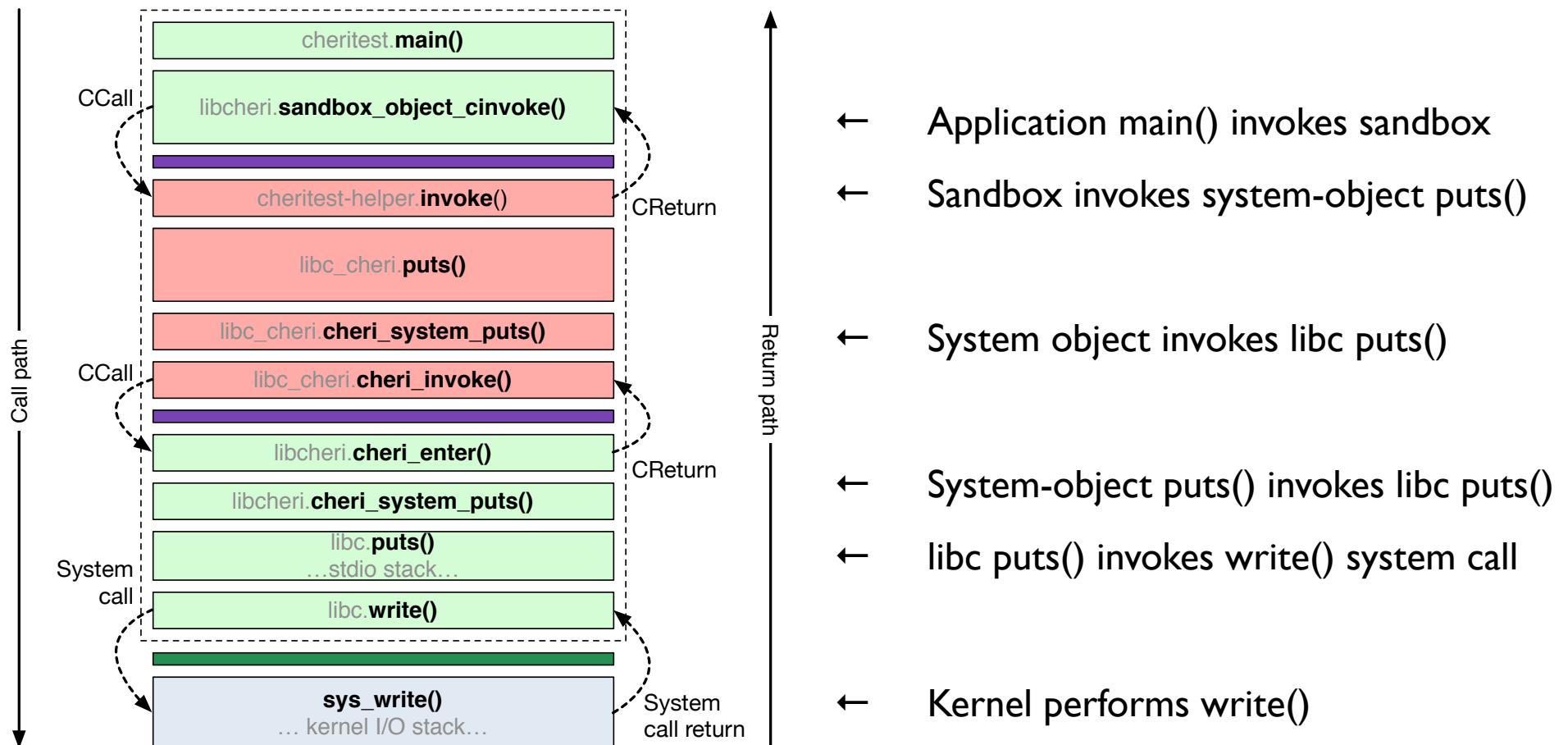
- CHERI register file preserved for each user thread
- CCall/CReturn exception handlers: object-capability invocation
- CHERI “trusted stack” for object-capability return path
- Sandbox fault recovery unwinds trusted stack on MMU fault, capability fault, or other thread traps
- Kernel accepts system calls only from in-process protection domains with ambient authority; requires using the system class
- Kernel debugging extensions for CHERI LLDB
- CHERI memory protection via CHERI Clang/LLVM
- libcheri(3) API to create and invoke sandboxes
- Extensions to the procstat(1) tool to track sandbox state



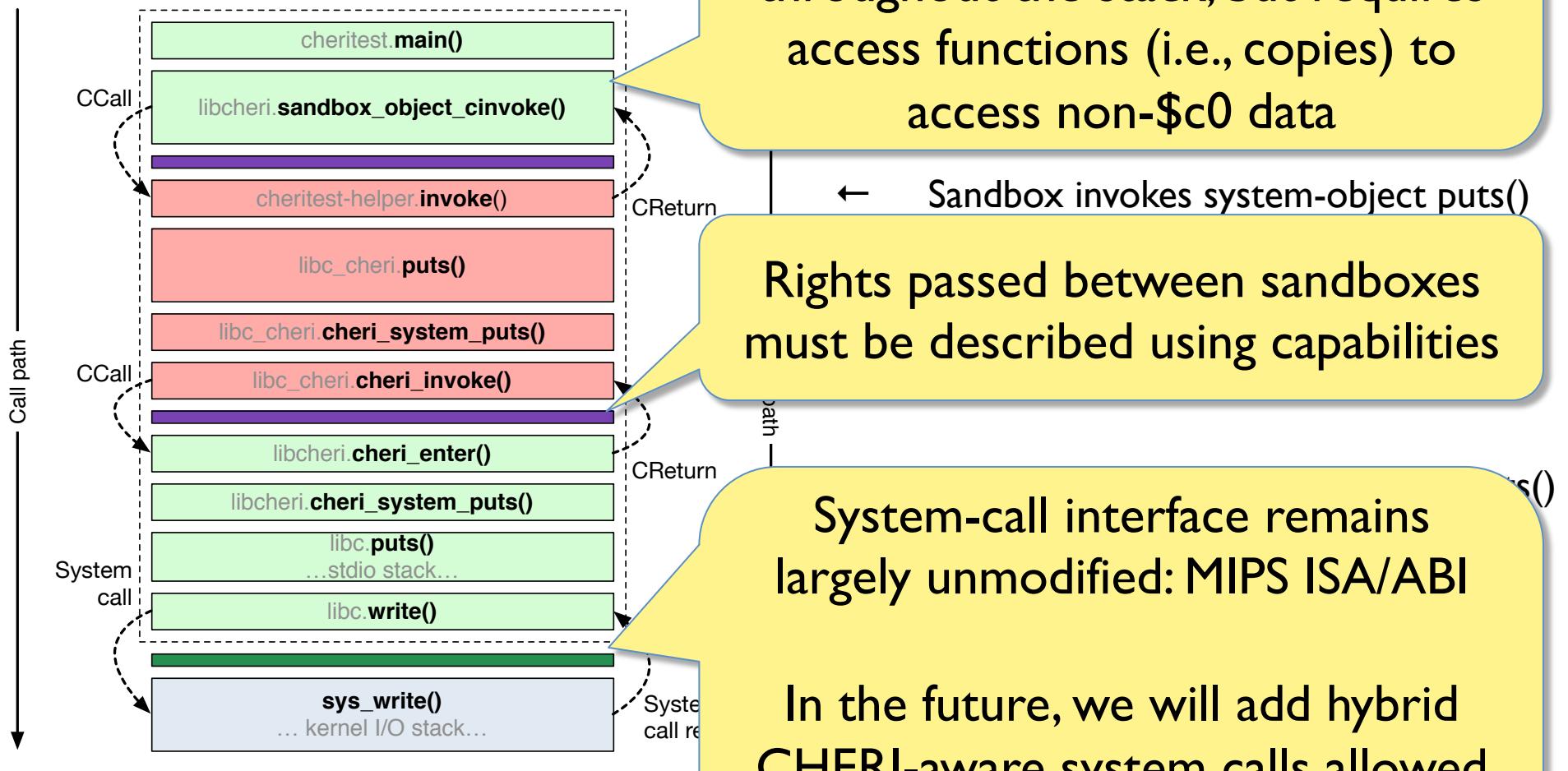
CHERI Clang/LLVM/LLDB

- CHERI Clang/LLVM
 - Clang supports new qualifiers, builtins for capability manipulation
 - LLVM CHERI code generation extends MIPS support
 - Pointers use MIPS representation and instructions by default
 - Pointers tagged as `_capability` generates CHERI instead of MIPS
 - Experimental CCured work automatically converts C code to CHERI ISA
- CHERI SDK
 - Complete cross-development environment: toolchain, libraries, headers.
- CHERI LLDB
 - LLDB supports CHERI registers in core files; live debugging in-progress

Example object-capability/sandbox invocation: “hello world”



Example object-capability/sandbox invocation



libcheri: object-capability sandbox API

- C-language bindings for CHERI object-capability sandboxes
 - Sandbox class
 - For now, memory image; soon, ELF binary (or segment)
 - new, method_declare, destroy
 - Sandbox object
 - Instantiated class with data
 - new, getsystemobject, cinvoke, destroy
 - Small assembly stubs for caller invoke() and callee enter()

NAME libcheri, sandbox_class_new, sandbox_class_method_declare, sandbox_class_destroy, sandbox_object_new, sandbox_object_getsystemobject, sandbox_object_cinvoke, sandbox_object_invoke, sandbox_object_destroy -- Library interface for CHERI sandboxing

LIBRARY library ``libcheri''

SYNOPSIS

```
#include <machine/cheri.h>
#include <machine/cheric.h>
#include <sandbox.h>

int
sandbox_class_new(const char *path, size_t sandboxlen,
                  struct sandbox_class **sbcp);

int
sandbox_class_method_declare(struct sandbox_class *sbcp, u_int methodnum,
                             const char *methodname);

void
sandbox_class_destroy(struct sandbox_class *sbcp);

int
sandbox_object_new(struct sandbox_class *sbcp,
                   struct sandbox_object **sbopp);

struct cheri_object
sandbox_object_getsystemobject(struct sandbox_object *sbop);
#endif __has_feature(capabilities)

register_t
sandbox_object_cinvoke(struct sandbox_object *sbop, u_int methodnum,
                      register_t a1, register_t a2, register_t a3, register_t a4,
                      register_t a5, register_t a6, register_t a7, capability_void *c3,
                      capability_void *c4, capability_void *c5, capability_void *c6,
                      capability_void *c7, capability_void *c8, capability_void *c9,
                      capability_void *c10);

#else

register_t
sandbox_object_invoke(struct sandbox_object *sbop, u_int methodnum,
```



libc_cheri: sandboxed C library; libcheri system class

- Subset of key C functions
 - Useful functions useable without ambient authority (e.g., snprintf)
 - Bottom-end functions invoke CHERI system-class object capabilities instead of system calls
- Kernel rejects calls without ambient authority
 - Sandboxes must request operations with ambient effects through CHERI system class

procstat(1): sandbox monitoring

```
% slogin -i .ssh/id_cheri_host ctsrd@cheritest.sec.cl.cam.ac.uk
Last login: Sat Nov 16 03:26:50 2013 from ip-64-134-230-112.public.wayport.net
FreeBSD 11.0-CURRENT (CHERI_DE4_SDROOT) #8 825c7e7(master)-dirty: Sat Jan 11 00:35:25 GMT 2014
```

```
% procstat -RX 7114
```

| PID | COMM | CLASS | METHOD | INVOKE | FAULT | SMIN | SMAX | SMEAN | SMEDIAN |
|------|-----------|----------------------|------------|--------|-------|--------|--------|--------|---------|
| 7114 | cheritest | cheritest-helper.bin | md5 | 4 | 0 | 10116 | 158925 | 47478 | 10436 |
| 7114 | cheritest | cheritest-helper.bin | abort | 1 | 1 | 3187 | 3187 | 3187 | 3187 |
| 7114 | cheritest | cheritest-helper.bin | helloworld | 1 | 0 | 452296 | 452296 | 452296 | 452296 |
| 7114 | cheritest | cheritest-helper.bin | puts | 1 | 0 | 456118 | 456118 | 456118 | 456118 |
| 7114 | cheritest | cheritest-helper.bin | syscall | 1 | 0 | 6551 | 6551 | 6551 | 6551 |
| 7114 | cheritest | cheritest-helper.bin | divzero | 3 | 3 | 2900 | 3166 | 3005 | 2950 |
| 7114 | cheritest | cheritest-helper.bin | malloc | 0 | 0 | 0 | 0 | 0 | 0 |

- Libcheri exports statistics on sandbox classes, objects, and methods
- libprocstat(3) and procstat(1) can query/print this
- libprocstat(3) provided data backend for demo UI

In progress: open sourcing CHERI

- Complete open-source hardware-software research/teaching stack
- BERI Open Systems CIC (“Community Interest Company”) Dec 2013
- BERI Apache-style license (HW), BSD license (SW)
- Physical designs for DE4 tablet, interconnect boards
- CHERI and CHERI2 Bluespec designs; debugging components/tools
- CHERI test suite, formal models
- FreeBSD device drivers
- CheriBSD capability support
- CHERI Clang/LLVM/LLDB
- ETA: January/February 2014



CHERI next steps

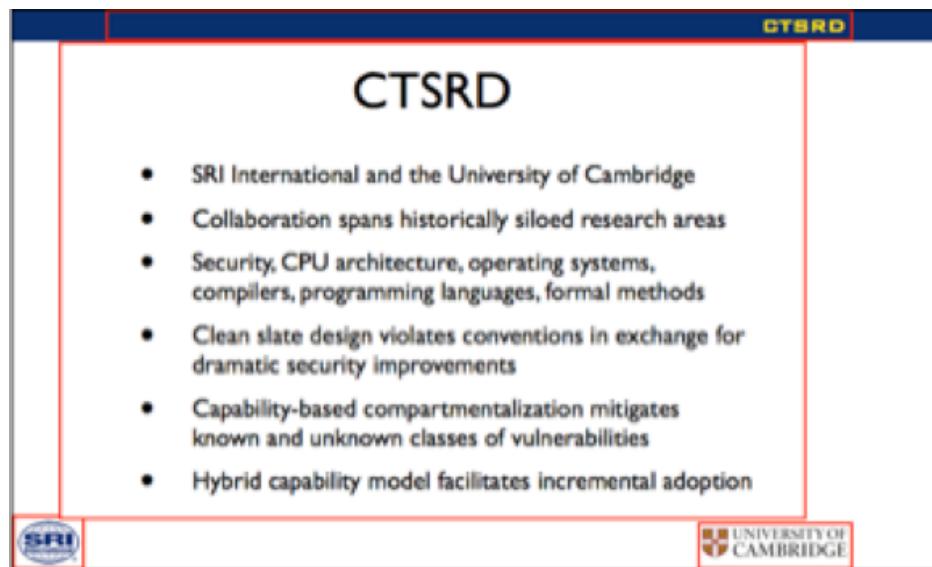
- CheriBSD kernel features (e.g., debugging, lazy switching)
- “Pure” CHERI ISA support for Clang/LLVM
- CHERI LLDB full feature support
- CCured-like automated use of memory protection
- Further CHERI ISA refinements: e.g., explicit CNULL
- Shift stack, heap access to CHERI ISA
- CCall/CReturn hardware optimizations
- Linker support for capabilities
- CHERI multithreading/multicore
- Additional languages: Object C, Ocaml

Compartmentalized packet capture and processing

CHERI DEMO



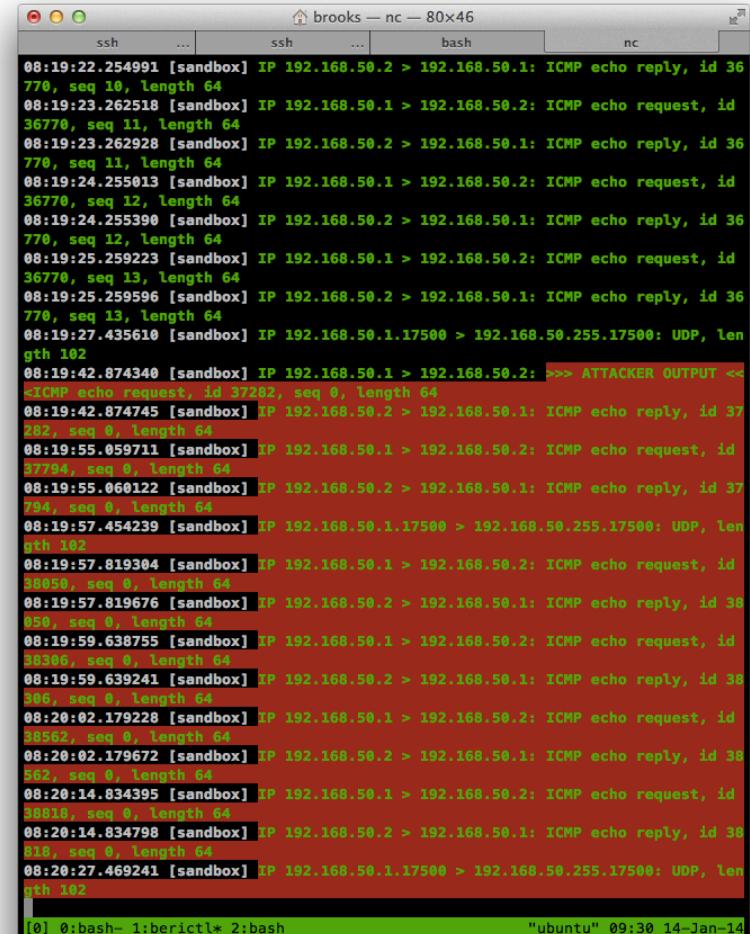
November 2012 - CheriPoint



- ✓ Bespoke compartmentalized CHERI presentation package
- ✓ Sandboxing mitigates trojan inserted in PNG library
- ✗ Largely MIPS ISA code generated from C
- ✗ A small amount of utility code written in CHERI assembly
- ✗ Static sandboxing policy

CHERI tcpdump demonstration

- Memory protection + compartmentalization
 - OS support for CHERI thread contexts
 - Compiler capability pointers
 - Userspace libcheri sandboxing model
 - Compartmentalized packet printing
- Key results:
 - Applicability of hybrid capability model
 - Tight C-language/capability integration
 - Tradeoffs policy/performance/mitigation
 - Compartmentalization scalability
 - Variable granularity



```

brooks — nc — 80x46
ssh ... ssh ... bash nc
08:19:22.254991 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 36
770, seq 10, length 64
08:19:23.262518 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
36770, seq 11, length 64
08:19:23.262928 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 36
770, seq 11, length 64
08:19:24.255013 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
36770, seq 12, length 64
08:19:24.255390 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 36
770, seq 12, length 64
08:19:25.259223 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
36770, seq 13, length 64
08:19:25.259596 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 36
770, seq 13, length 64
08:19:27.435610 [sandbox] IP 192.168.50.1.17500 > 192.168.50.255.17500: UDP, len
gth 102
08:19:42.874340 [sandbox] IP 192.168.50.1 > 192.168.50.2: >>> ATTACKER OUTPUT <<
: ICMP echo request, id 37282, seq 0, length 64
08:19:42.874745 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 37
282, seq 0, length 64
08:19:55.059711 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
37284, seq 0, length 64
08:19:55.060122 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 37
284, seq 0, length 64
08:19:57.454239 [sandbox] IP 192.168.50.1.17500 > 192.168.50.255.17500: UDP, len
gth 102
08:19:57.819304 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
38050, seq 0, length 64
08:19:57.819676 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 38
050, seq 0, length 64
08:19:59.638755 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
38306, seq 0, length 64
08:19:59.639241 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 38
306, seq 0, length 64
08:20:02.179228 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
38562, seq 0, length 64
08:20:02.179672 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 38
562, seq 0, length 64
08:20:14.834395 [sandbox] IP 192.168.50.1 > 192.168.50.2: ICMP echo request, id
38618, seq 0, length 64
08:20:14.834798 [sandbox] IP 192.168.50.2 > 192.168.50.1: ICMP echo reply, id 38
618, seq 0, length 64
08:20:27.469241 [sandbox] IP 192.168.50.1.17500 > 192.168.50.255.17500: UDP, len
gth 102
[0] 0:bash- 1:bericctl* 2:bash
"ubuntu" 09:30 14-Jan-14

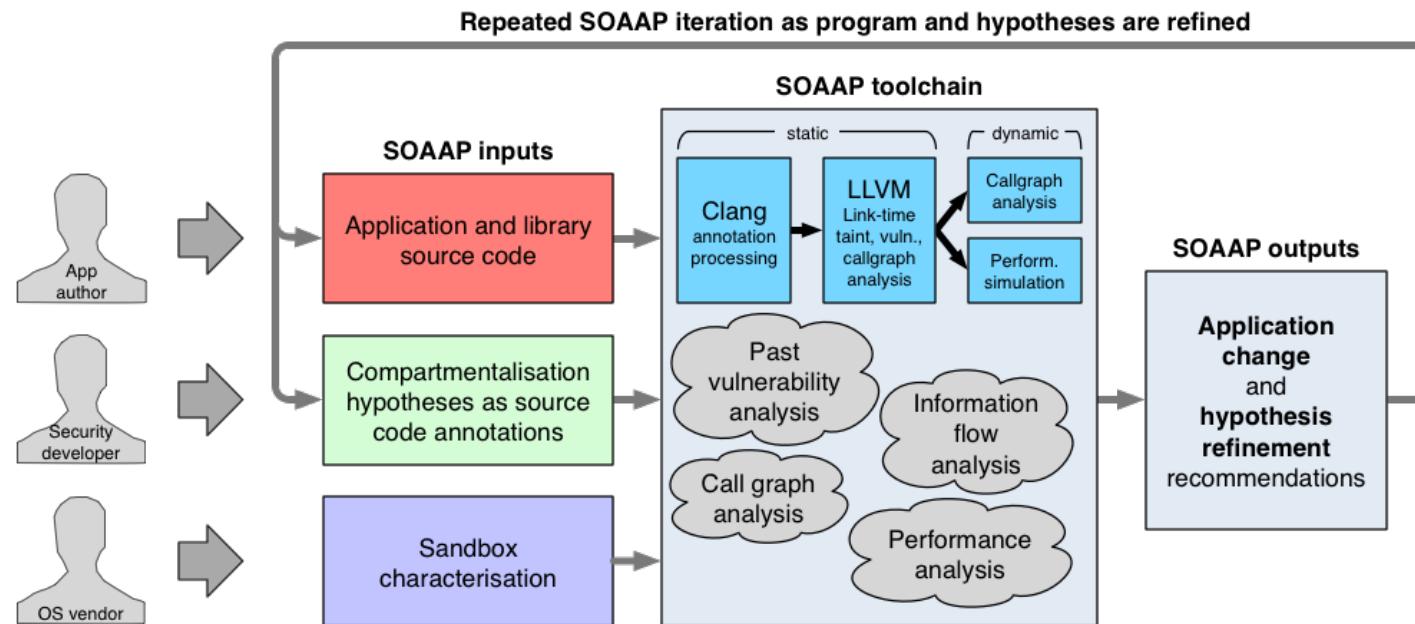
```

Software analysis and transformation

SOAAP AND TESLA



Security-oriented analysis of application programs (SOAAP)



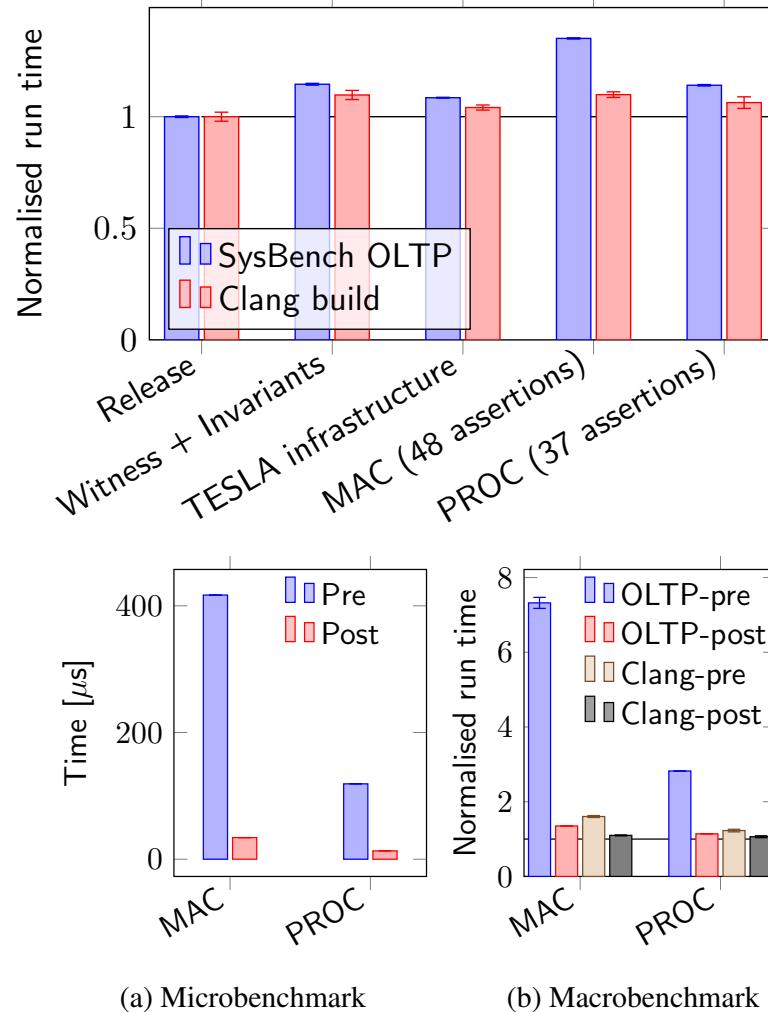
- Static and dynamic analysis tools to assist programmers when compartmentalizing applications
- Come see demo at poster session!

TESLA

- Pragmatic validation of run-time security properties
- LTL-like assertions embedded in code
- Compiler-generated instrumentation
- Significant outreach to potential open-source and corporate consumers
- Come see demo at poster session!



TESLA since last time



- Applied TESLA to OpenSSL, FreeBSD, Objective-C
- Found subtle bugs that eluded traditional debug tools
- Build cost: rebuilds less incremental
- Significant runtime cost optimizations

Conclusion

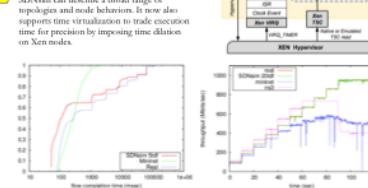
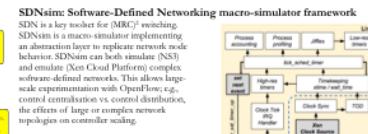
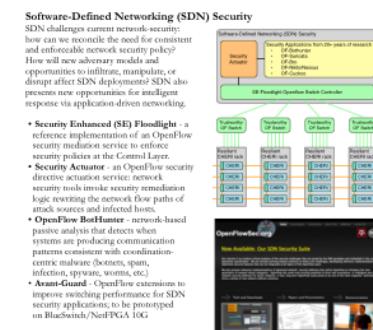
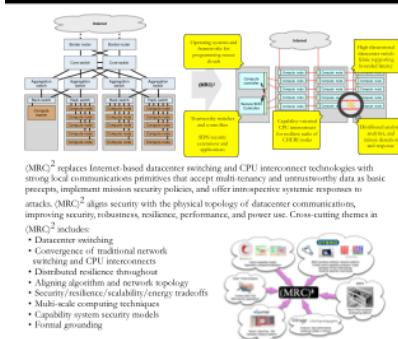
- Three years into the five-year project
- Mature CHERI hardware platform
- CheriBSD operating system
- CHERI Clang/LLVM/LLDB/SDK
- CHERI application exploration in progress
- SOAAP and TESLA tools maturing
- Smten, architectural extraction, and formal ISA models bearing early verification results

(MRC)² sister project

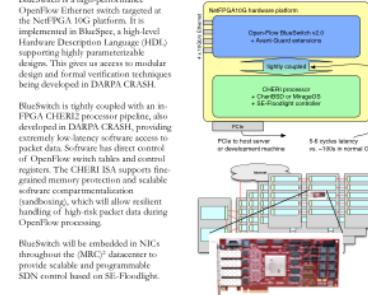


Modular Research-based Composably trustworthy Mission-oriented Resilient Clouds

Peter G. Neumann, Suresh W. Moon, Robert N. Watson, Nitin Dara, Brooks DeLine, Martin Fong, Matthew P. Giacomini, Kristen Gallie, Asif Khan, Myron King, Jong Hui Kim, Steven M. Hess, Alex Horvath, Patrick Lachaud, And Matuszewski, Bas Maessen, Thor Matthes, Andrew W. Moore, Alaa Mousa, Prashanth Mundkur, Robert Neutze, Philipp Peters, Gérald Rethoré, Charalambos Romanos, Hanan Saad, Mahrus Schaefer, Richard Uller, Jonathan Woodruff, Vusal Yigitcanlar, Dongri Yu, Björn A. Ziehl



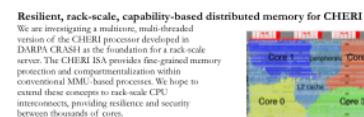
BlueSwitch: trustworthy resilient switches and switch controllers



NetFPGA10G and CHERI-NetFPGA10G
We are pushing NetFPGA10G
forward to enable high performance trusted
programmable switch controllers.

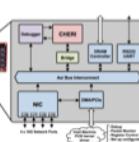
Recently prototyped CHERI processor platform from Altera based board platform to Xilinx based NetFPGA platform including:
• Avalon memory bridge
• AXI DRAM controller
• PCIe based debug interface
• RS232 UART console
• Network interface integration

Status: Booting CHERI OS on NetFPGA10G
Next: Network driver and further refinement
Significant contributions to the NetFPGA10G release:
• Reference network interface card
• Reference simple Ethernet switch
• Reference IPv4 router
• Leading NS-3 and NetSim funded outreach activities



The capability model provides CHERI with unprivileged information about software behaviour, which will allow us to implement local memory behaviour. We also hope to explore convergence between conventional memory design and local-area Ethernet switching technology, blurring the boundary between two very different design genres. Another CHERI feature is the low cost of object-capability invocation, and experimentation with alternative execution models.

We are refining multithreaded and multiversion of CHERI. We are also bringing up the CHERI operating system on multicore CHERI. Then we will exploit reliable FPGA-to-FPGA link technology developed under an EPSRC funded project to prototype rack-scale systems.



Secure cloud operating system
The (MRC)² operating system projects need communications and security facilities both within and between individual "compute" clouds. To exploit these facilities, we require new programming and communication models, encapsulated in operating systems and distributed computation frameworks. We are developing three frameworks: MirageOS, and OCaml-based OS grounded in the Cuckoo design philosophy; DIOS, a data-intensive OS for datcenters, and cluster extensions to CHERI-SQ, our adaptation of the FreeBSD OS to the CHERI CPU architecture. We are targeting distributed computation in a multi-tenant cloud fabric throughout.

MirageOS
Unikernels are single-purpose appliances that are compile-time specialised into standalone kernels, and scaled against modification when deployed to the Xen cloud platform. MirageOS, a library OS written in OCaml, provides a mechanism to build distributed units of work in a very compact way using a unikernel approach. We will also specialised Mirage-based switch control software to BlueSwitch nodes.



DIOS - A Data-Intensive Operating System
New work exploring how we can use a data centre as one distributed fault-tolerant real-time computer.

Our emphasis is on using capabilities to seamlessly straddle hardware, software and network domains.

CheriSD cluster extensions
The CHERI CPU architecture supports fine-grained address space memory protection and compartmentalization. As CPU interconnects increase to rack scale, providing communication within the rack will improve robustness and limit the impact of compromise. CHERI-SQ must be adapted to support multi-instance operation based by alternative communication substrates – initially, using the distributed memory interconnect to emulate a shared memory, and later, to support a distributed memory architecture interactions enabled by local capability invocations. This will require new OS and programming language abstractions, based on CHERI and the CHERI Cang/LLVM suite. Work will begin on CHERI-SQ extensions as the BIMPA inter-FPGA interconnect comes online linking cache-enhanced multi-core CHERI nodes.

RDD2 Networking
Resilient real-time data delivery (RDD2) network architecture is exploring buffered and switchless network architecture focused on bounded latency. Bounded latency improves fault-tolerance, elections and query performance. For 16 cores per node, our pmt achieves end-to-end latencies of under 75µs with 99.999% probability.



- Heavy use of CTSRD-derived CHERI
- Multithreaded and multicore CHERI prototypes
- CHERI on NetFPGA 10G



Q&A



CTSRD at the PI Meeting



Dr Peter G.
Neumann



Dr Robert N. M.
Watson



Dr Simon W.
Moore



Dr Jonathan
Anderson



Dr David
Chisnall



Dr Nirav
Dave



Mr Brooks
Davis



Mr Rance
DeLong



Dr Khilan
Gudka



Dr Theo A.
Markettos



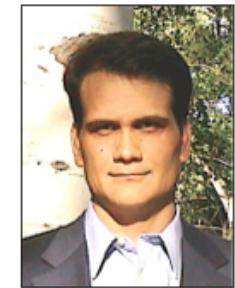
Mr Ed
Maste



Dr Michael
Roe



Mr Colin
Rothwell



Mr Stacey
Son