CTSRD Project Briefing

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

DARPA CRASH PI Meeting
Jacksonville, Florida USA
24 September 2014
http://www.cl.cam.ac.uk/research/security/ctsrd/
CTSRD team at the PI Meeting

Dr Peter G. Neumann
Dr Robert N. M. Watson (in absentia)
Prof. Simon W. Moore
Dr David Chisnall
Dr Nirav Dave
Mr Brooks Davis
Dr Khilan Gudka
Dr Andrew W. Moore
Dr Prashanth Mundkur
Dr Michael Roe
Mr Stacey Son
Dr Jonathan Woodruff

... and a large team of collaborators at SRI, Cambridge, and Bluespec contributing to hardware, software, security, formal methods, engineering, etc!
Motivation: Vulnerability Crisis

- Constant attacks on critical infrastructure and private industry
- New vulnerabilities and exploits but more importantly:
  - New vulnerability classes, exploit techniques
  - Threat (and exploitation) of software supply-chain attacks
- Total asymmetry between attacker and defender: ‘just one bug’
Fundamentals

We want security based on the

Principle of **least privilege**

Easily said but hard to deliver
Multipronged strategy

1. Eliminate fragile properties in the execution substrate (ISA/compiler)

2. Support scalable, granular compartmentalization of applications at a logical level (programmer-defined objects)

3. Build realistic hardware and software artifacts allowing structured evaluation of security, performance, etc.

4. Develop new tools to analyze higher-level security properties, project into compilation and execution

5. Develop and judiciously apply new formal techniques to hardware and software
Progress since January 2014 (1 of 3)

- ISCA paper presented on CHERI ISA v2: The CHERI capability model: Revisiting RISC in an age of risk.
  Other 2014 papers at AsiaBSDCon, Eurosyst (TESLA), OOPSLA (Smten), SIGCOMM (NetStackSpecialization)

- CHERI ISA v3: Stronger C-language support, multiple ABIs, transparent use of capabilities
  - Benefitting from large application experience
  - Improvements at every level of the stack (apps, OS, compiler, processor)
Progress since January 2014 (2 of 3)

• Toolchain and OS provide increased support for compartmentalization
  • Maturing intra-process compartmentalization: object model, fault handling, etc.
  • Improved security analysis tool (SOAAP)
  • Trial compartmentalizations of large applications such as NetSurf on CHERI
Progress since January 2014 (3 of 3)

• Platform maturity
  • Improvements to the CHERI ISA
  • L3 formal ISA model of CHERI & more regression tests
  • Dual-core CHERI (with help from our MRC project)
  • Software platform: USB support, X11, ...

• Formal methods: SRI formal analysis tools embedded in BSV compilation, Smten as a developer-friendly front end, and architectural extraction to simplify analysis of subsets of specifications.
MIT Lincoln Laboratory is Evaluating CHERI

Demo at poster session
PROJECT REVIEW
In the beginning: Capsicum

- Software-based hybrid capability model: OS APIs for application compartmentalization (USENIX Security 2010)
- Joint Cambridge/Google project that continues
- Experimental feature in FreeBSD 9.x; shipped out-of-the-box in FreeBSD 10.0
- Ongoing FreeBSD Foundation, Google funding
  - Growing number of FreeBSD programs that use 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, patches submitted to lkml for review
CHERI Architectural Concepts

• Incrementally deployable hybrid design:
  • Composes cleanly with conventional MMU-based protection and operating systems
  • ISA, ABI compatibility support a blend of language/compiler models
  • Can invest first in most trusted (TCBs), least trustworthy software

• **Memory capabilities:** least privilege for ISA-level data access

• **Object capabilities:** least privilege for programmer-defined objects
First steps (2011)

- Early and buggy MIPS-ISA-derived CPU; no MMU
- Early prototype CHERI ISA and capability coprocessor
- Small single address-space microkernel uses memory capabilities for isolation
- Simple, hand-crafted application with lots of assembly to demonstrate sandboxing
- Proof of concept, but little more
Getting better (2012)

- CHERI processor mature enough to run FreeBSD OS
- Basic capability support in FreeBSDâ‡”CheriBSD prototype
- Userspace apps able to use capabilities for memory protection, basic sandboxing
- Compartmentalized slide-viewer application
  - Little C capability support; relied heavily on MIPS hybridization
  - Mitigated software supply-chain trojan
- Object model under development
Toward real applications (2013)

• Compiler supports manually annotated __capability pointers
• OS now has CHERI thread contexts, object-capability invocation, more mature userspace sandboxing runtime
• tcpdump demo shows tight C-language/capability integration
• CHERI cloud server for remote development
• SSH into live CHERI system; off-the-shelf applications
Real applications (2014)

- CHERI ISA v3 converges capability and fat-pointer models by adding offsets to capabilities
- Compiler can compile pointers as capabilities
- OS maps protection/security-model faults to UNIX signal delivery
- Open-source web client (Netsurf) ported with sandboxed rendering
- Early multicore support
Netsurf compartmentalization with Squirrelmail e-mail client

Capsicum process-level compartmentalization

Folder-list i-frame

CHERI object

Message-header i-frame

CHERI object

Message-body i-frame / CHERI object

PNG rendering

CHERI object

Under the hood: HTTP, TLS, ... objects
Toward productization (2015)

• Contract extension into 2015: “CHERI PIE”
• Mature OS/class-library compartmentalized applications, capability/sandboxing-aware debugger, etc.
• More real-world application experience: web browser, e-mail client
• Refine software tools: compiler and analysis (SOAAP and TESLA)
• Evaluate security/performance/compatibility trade-offs at scale
• Engage with industry & academia
• Engage open-source hardware and software communities via BERI Open Systems CIC – both research platform, and CHERI
• Extend formal analysis and grow regression tests
• Explore smaller (128b) capabilities, to reduce cache footprint and increase opportunities for potential tech transfers
ISA/COMPILER CO-DESIGN
Real application experience

- Real application experience needed to understand compiler requirements
- Real compiler (Clang/LLVM) needed to understand ISA requirements
- C programmers do many evil things with pointers!
- Original capability model provided only protection & sandboxing: (tag, permissions, type, base, limit)
- CHERIv3 fat-pointer-style capabilities are full pointer replacements: (tag, permissions, type, base, limit, offset)
- Many lessons learned from recent fat-pointer research: Softbound, Hardbound, CCured, Low-Fat Pointers (SAFE), etc.
Return Address Protection for Free

• Program counter was an offset relative to the program counter capability (PCC)
• Offset is now integral property of PCC
• Capability jump (return) / jump-and-link (call) no longer need separate offset
• Safer returns: return address is now a capability, making ROP attacks much harder; also, no data-store instruction can fabricate a return address
Results from porting tcpdump

• 3006 (4.5%) lines of code changed to add __capability qualifiers (could be automated)

• 1577 (2.4%) lines changed for CHERIv2, quite a few of them subtle changes, to make pointers work as capabilities – Not needed with CHERIv3

• 2 lines changed for CHERI v3 to simplify bounds checking to use the hardware

• Great progress toward minimal changes even to unsafe languages like C
A tale of 2 3 ABIs

- Incremental deployment is vital for testing
- Rewriting (or even recompiling) all code at once isn’t practical

More compatible: n64 Pure MIPS

More safe: n64 + CHERI some pointers are capabilities

Sandbox: all pointers are capabilities
Compartmentalization Requirements

• Memory protection for isolation
  • provided by capabilities

• A mechanism for calling between isolated domains
  • provided by object-capability invocation

• Optimization is for frequent and high-volume intra-application communication (unlike MMUs)
CheriBSD Object-Capability Model

• Kernel implements exception handlers for CCall and CReturn instructions
  • Constructs a secure return path via trusted stack
  • An implementation of strong mutual distrust
• Security/protection faults mapped to sandbox unwind or UNIX signals (e.g., to userspace sandbox or language runtime)
• ‘Sandboxing’ is a useful compartmentalization design pattern; many other patterns supported
  • E.g., mutual rather than asymmetric distrust
Object-capability/sandbox invocation

- Application code with ambient authority
- Sandboxed code
- libc stubs
- libc with ambient authority
- Kernel I/O stack
ENGINEERING UNDERPINNINGS
Debug & Regression Testing

- LLDB debugger now includes initial capability support
- Simulation and hardware tracing
  - trace comparison
- Automatic regression test system
Formal Modeling

- **New:** L3 formal model of CHERI ISA incl. dual-core
  - The first formal model to boot a commodity OS
  - Trace comparison on single and dual-core CHERI
  - Found system bugs
- First steps taken towards ACL2BSD: FreeBSD on ACL2 theorem prover’s x86 model
  - Possible direction: CHERI on x86?
  - In collaboration with Warren Hunt, UT Austin
Formal Methods

• Formal description of capabilities and their security properties in SRI’s SAL model checker

• Architectural extraction tool developed
  • Transform complex pipelined processor from HDL representation into an architectural description to be checked

• Smten tool developed to help: automatic translation of high-level symbolic computations into SMT queries
Security-oriented analysis of application programs (SOAAP)

- Static and dynamic analysis tools assist programmers when compartmentalizing applications
SOAAP since last time

• Focus on applying SOAAP to already compartmentalized programs, such as Chromium, to be able to evaluate and validate sandboxing policy

• Numerous optimizations to enable analyses of these very large code bases

• Annotations for modeling RPC communication

• SOAAP now takes a sandbox platform description and can provide relevant feedback:
  • Capsicum and Seccomp initially
SOAAP Directions

• Continue developing SOAAP so as to be able to handle real-world large complex applications:

  • Enrich annotation language to capture required/implemented compartmentalization policies

  • Further develop sandbox platform descriptions to evaluate security across multiple platforms

  • Extend performance emulation to more accurately evaluate performance tradeoff

• Program visualization to aid programmer understanding of program structure and communication patterns, as a result of sandboxing
TESLA

- Runtime validation of temporal security properties
- LTL-like assertions embedded in C code
- Compiler-generated instrumentation
- Interaction via DTrace, debuggers
- Significant outreach to potential open-source and corporate consumers
- Paper at EuroSys 2014
- TESLA assertions are potential inputs to ACL2BSD
NEW TASK: Predictably Trustworthy Whole-System Hardware Integration
New Task: Predictably Trustworthy Whole-System Hardware Integration

• This new task began in June 2014.

• Modern computer systems (e.g., laptops and mobile devices) contain many microprocessors, often with direct memory access to the main processors.

• We are seeking total systems (including microprocessors) with significantly improved overall trustworthiness.

• Thus far we have early versions of
  • A catalog of known input-output vulnerabilities
  • A taxonomy of vulnerabilities
  • Requirements for trustworthy system architectures
  • What a CHERI-like extended architecture might be
WRAP-UP
Summary

• Four years into the five-year project
• Mature CHERI hardware platform
• CheriBSD OS with in-process memory/object-capability model
• Maturing compiler support (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
CTSRD Project Open Source

- BERI/CHERI processor designs and tools
- Terasic DE4 CHERI tablet physical build
- CheriBSD operating system + applications
- CHERI Clang/LLVM compiler suite
- SOAAP toolchain
- TESLA toolchain + TESLABSD
- Smten SMT toolchain
- Capsicum in FreeBSD 10; Google’s patches for Linux
- RSN: Sandstorm clean-slate network stack (SIGCOMM)
- http://www.cl.cam.ac.uk/research/security/ctsrd
Conclusions

- Complete stack needed for real evaluation
- Almost-C is not good enough!
- Protection model provides the basis for object capabilities
  - Also mitigates many common exploit techniques
- Object capabilities support effective compartmentalization
  - Strong mitigation for many classes of vulnerabilities
  - Augmenting sandboxing with other design patterns
- Further advances in tools needed to compartmentalize large code bases
- Results so far are compelling for industrial transition
Q&A
CTSRD team at the PI Meeting

Dr Peter G. Neumann
Dr Robert N. M. Watson (in absentia)
Prof. Simon W. Moore
Dr David Chisnall
Dr Nirav Dave
Mr Brooks Davis
Dr Khilan Gudka
Dr Andrew W. Moore
Dr Prashanth Mundkur
Dr Michael Roe
Mr Stacey Son
Dr Jonathan Woodruff

... and a large team of collaborators at SRI, Cambridge, and Bluespec contributing to hardware, software, security, formal methods, engineering, etc!