Advanced Topics in Computer Architecture

Secure Processors I: CHERI

Prof. Simon W. Moore

UNIVERSITY OF CAMBRIDGE

Computer Science & Technology

Background

- CHERI: secure processor design by Cambridge + SRI International
- Timely:
  - Big UK funding push to commercialise the technology:
    Industry Strategy Challenge Fund: **Digital Security by Design**
    - £70m UK government funding + £116m from industry
  - Started 26th September 2019
  - ARM making the Morello test chip and board platform to be shipped to partners Q1 2022
- Based on substantial research
  - 120+ engineer/research years of effort
  - >$24m of DARPA funding

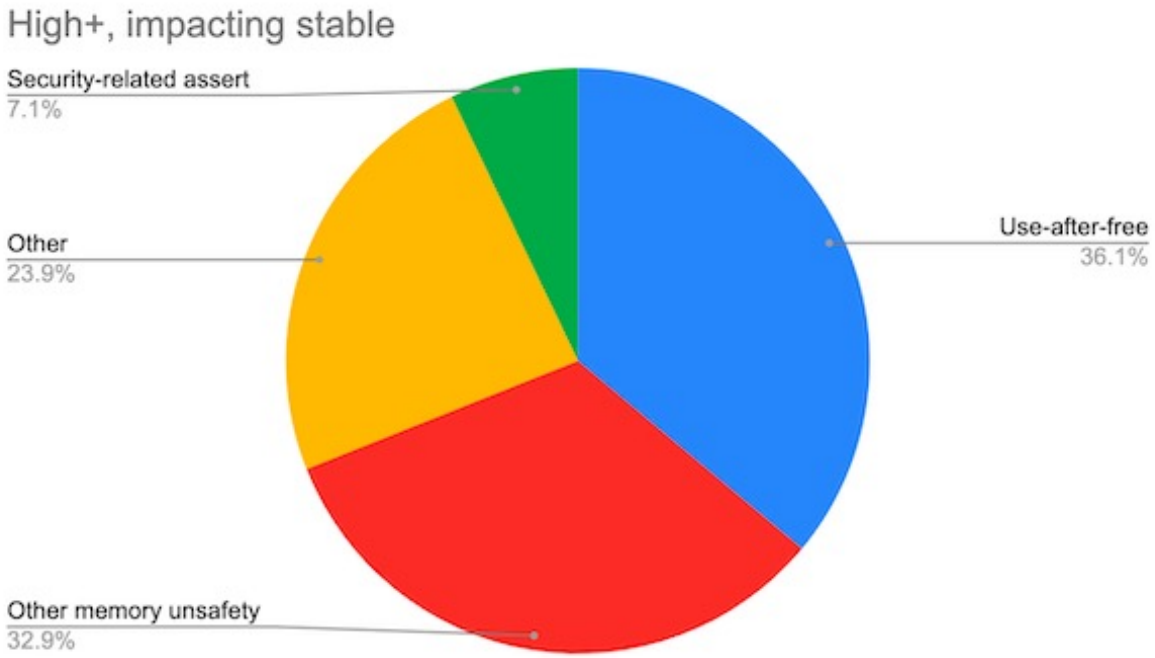
Motivation – Memory Safety

- Matt Miller (MS Response Center) @ BlueHat 2019:
  - From 2006 to 2018, year after year, 70% MSFT CVEs are memory safety bugs.
  - First place: spatial safety
    - Addressed directly by CHERI
  - Second place: use after free
    - Temporal memory safety is made efficient by exploiting CHERI capability validity tags to quickly and precisely find pointers during revocation

Motivation – Chromium Browser Safety

“70% of our serious security bugs are memory safety problems”

www.chromium.org/Home/chromium-security/memory-safety

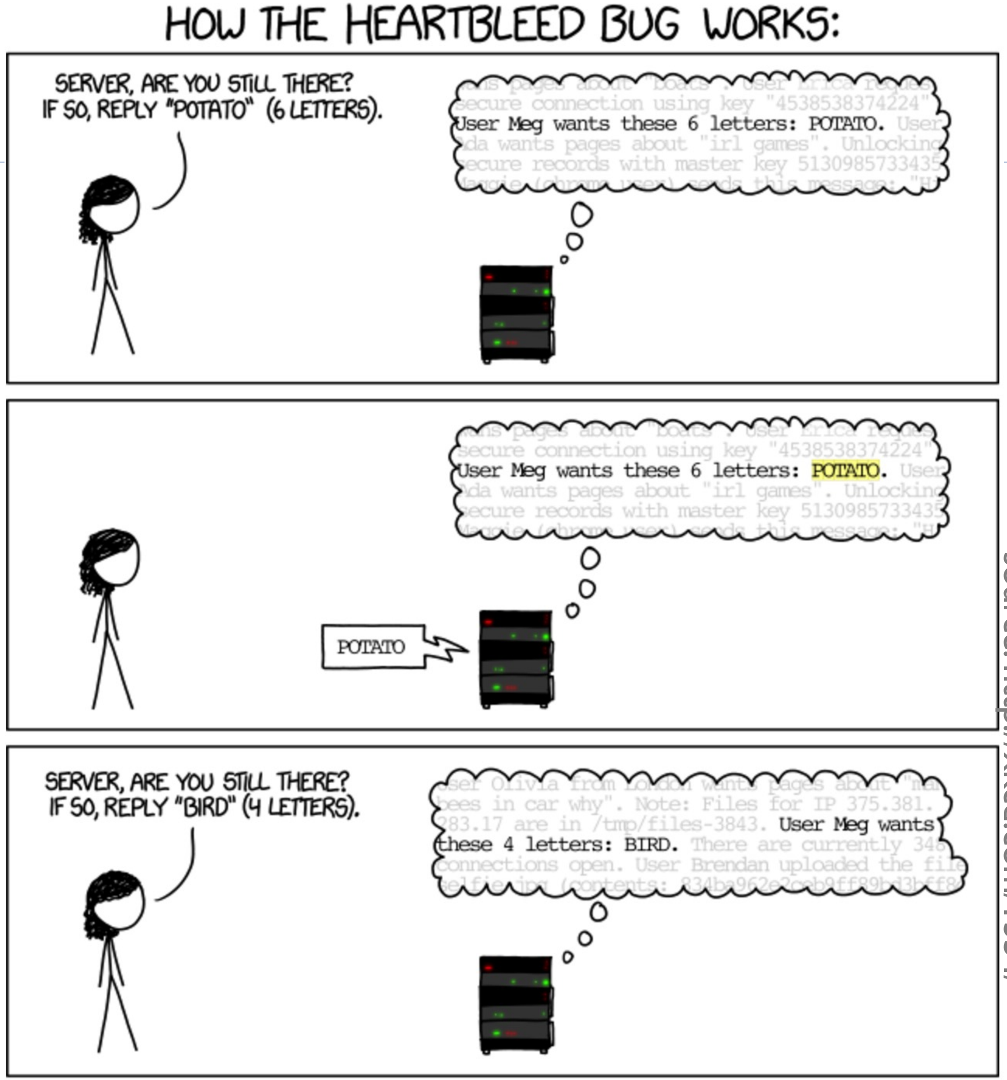


Motivation – The Eternal War in Memory*

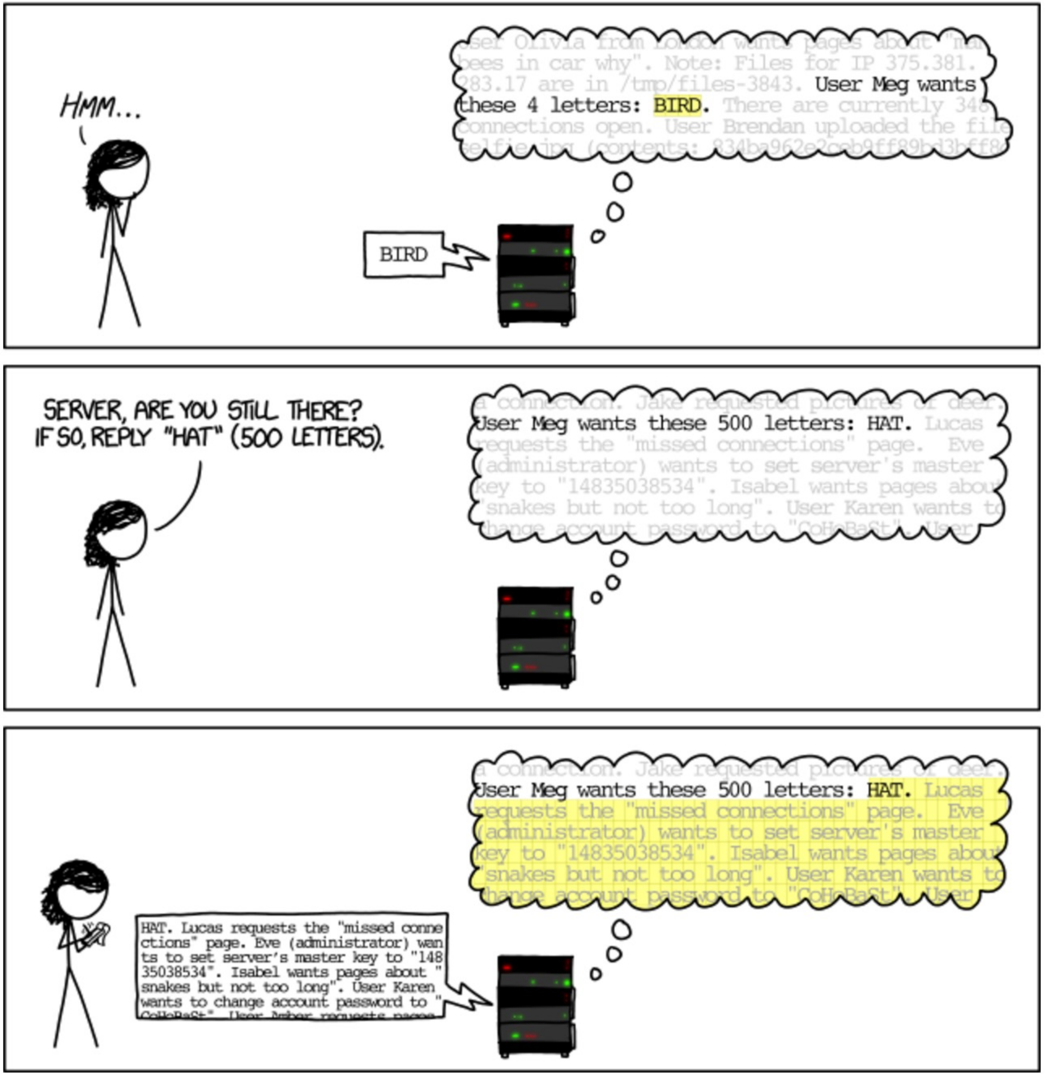
- Many security vulnerabilities exploit memory safety violations

* Title based on Oakland 2013 paper: SoK: Eternal War in Memory, László Szekeres, Mathias Payer, Tao Wei, Dawn Song

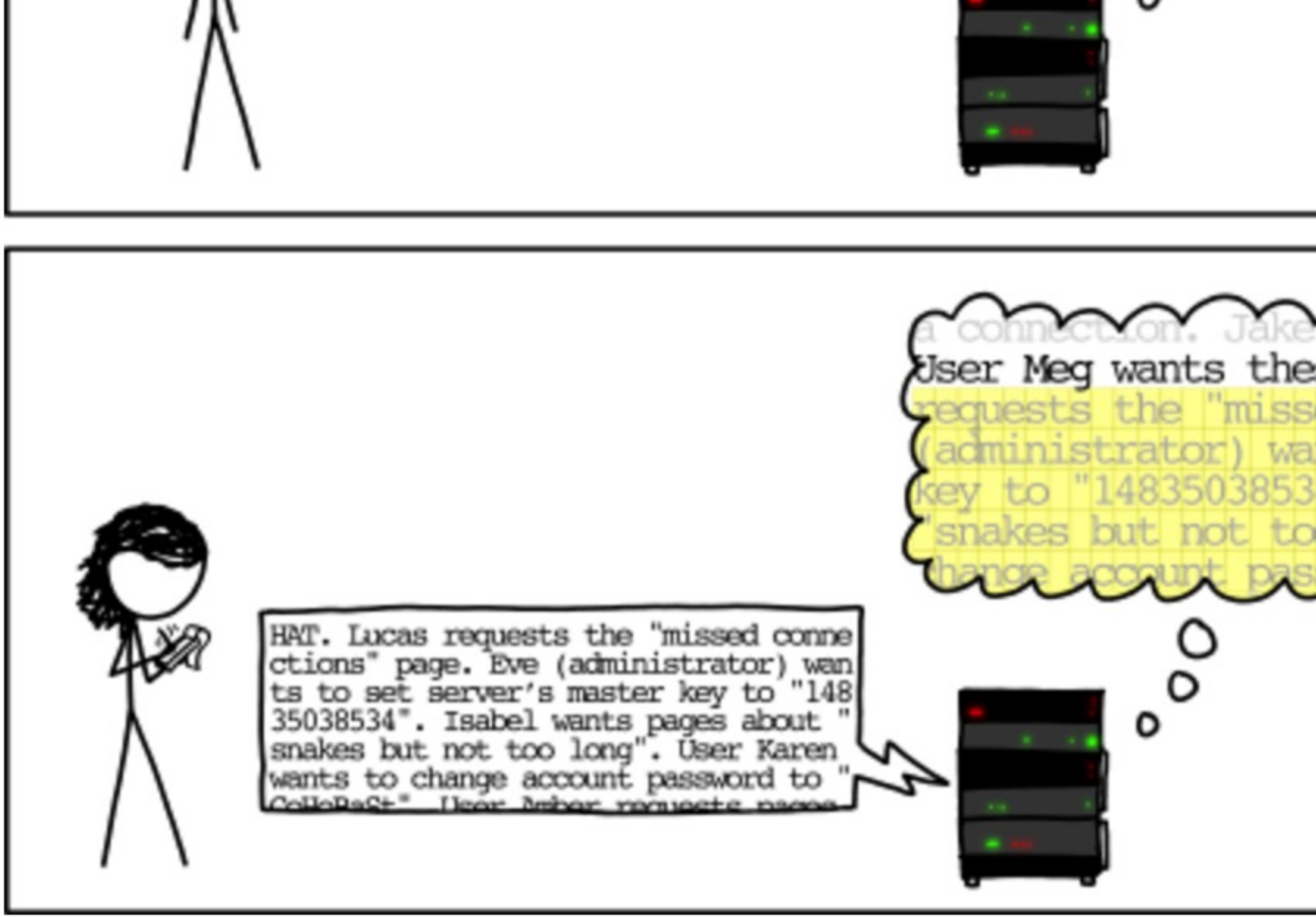
HeartBleed


HOW THE HEARTBLEED BUG WORKS:
SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
ans pages about "boats". User Erica reques secure connection using key "4538538374224". User Meg wants these 6 letters: POTATO. User Ada wants pages about "irl games". Unlocking secure records with master key 5130985733435 Meggie (chrome user) sends this message: "H
User Meg wants these 6 letters: POTATO.
POTATO
SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Olivia from London wants pages about "mad bees in car why". Note: Files for IP 375.381.283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 346 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8

source: http://xkcd.com/1354/


HMM...
User Olivia from London wants pages about "man bees in car why". Note: Files for IP 375.381.283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 348 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
a connection. Jake requested pictures of deer. User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Amber requests pages

source: http://xkcd.com/1354/


HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Amber requests pages
a connection. Jake
User Meg wants the
requests the "miss
(administrator) wa
key to "148350385
"snakes but not to
hange account pas

Went wrong? How do we do better?

- Classical answer:
  - The programmer forgot to check the bounds of the data structure being read
  - Fix the vulnerability in hindsight – one line fix:
    if (1+2+payload+16 **>** s->s3->rrec.length) **return** 0;
- Our answer:
  - Preserve bounds information during compilation
  - Use hardware (CHERI processor) to dynamically check bounds with little overhead and guarantee pointer integrity & provenance

CHERI Approach to Memory Safety

- The **principle of intentional use**
  - Ensure that software runs the way the programmer intended, not the way the attacker tricked it
  - Approach:
    - Guaranteed pointer integrity & provenance; efficient dynamic bounds checking
    - Instructions always accept capability operands, and never look them up automatically (unlike an MMU)
- The **principle of least privilege**
  - Reduce the attack surface using software compartmentalization
    - This mitigates known and unknown exploits
  - Approach: highly scalable and efficient compartmentalization
- Robust **deterministic protection**, not probabilistic debugging measures

Preserve RISC philosophy

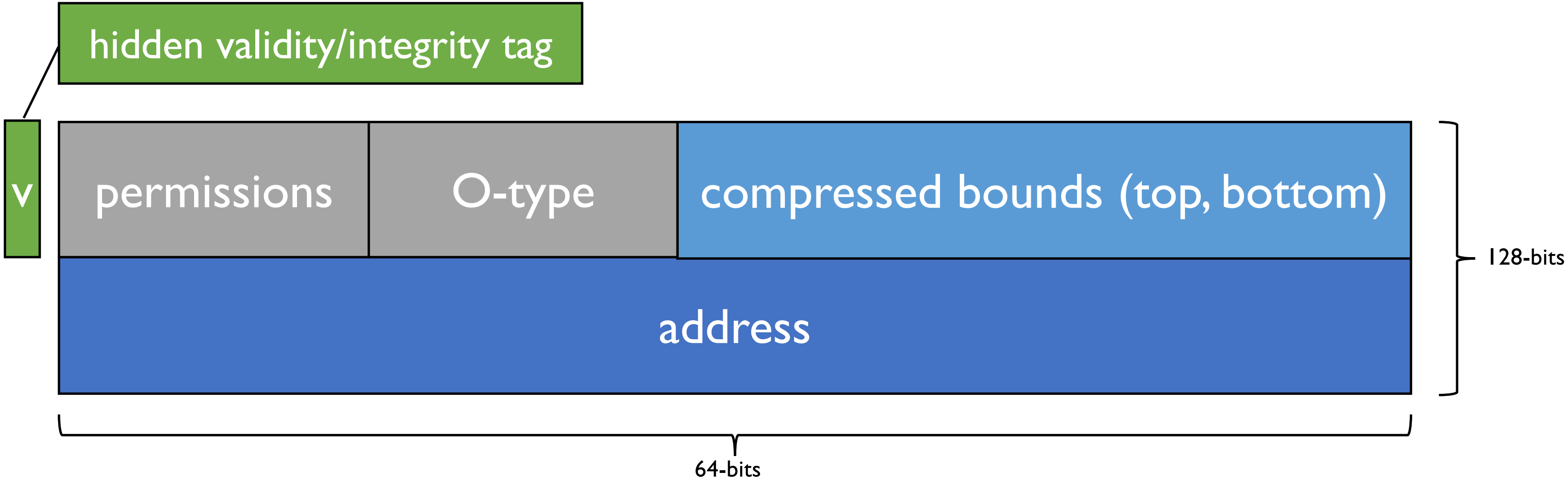
- “No” to:
  - Microcode
  - New table lookups
  - Exotic external memory
  - Extended pipeline
  - Reduction in clock frequency
  - Reduced addressing modes
  - Crypto (not needed here)

- “Yes” to:
  - Low-level hardware functionality on which many software structures can be built
  - Compiler friendly
  - Get the compiler, linker and run-time system to do much of the work, not the ISA+decoder
  - Keep it as simple as possible!

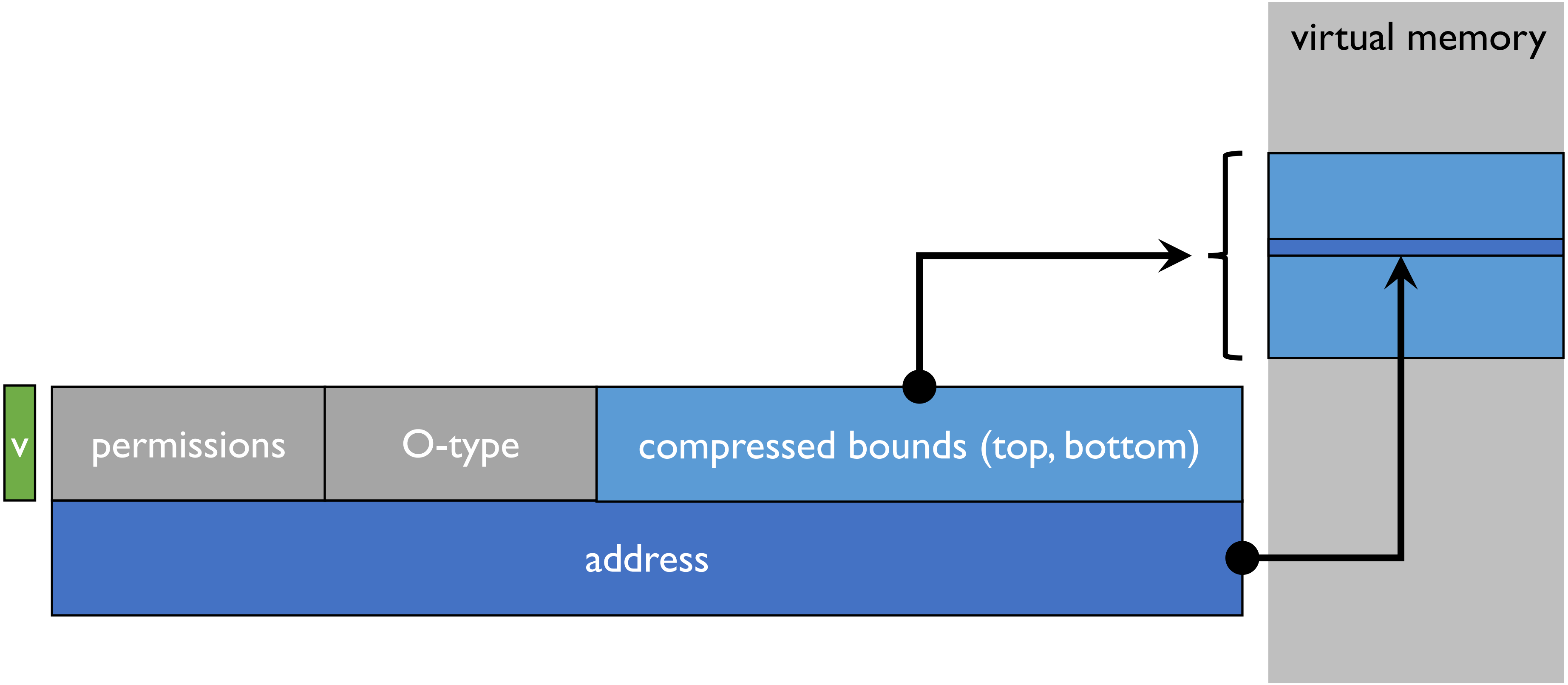
CHERI HARDWARE ARCHITECTURE

A new type – the **Capability**

- CHERI Capability = bounds checked pointer with integrity
- Held in memory and in (new or extended) registers


hidden validity/integrity tag
v
permissions
O-type
compressed bounds (top, bottom)
address
128-bits
64-bits

A new type – the **Capability**



New Instructions

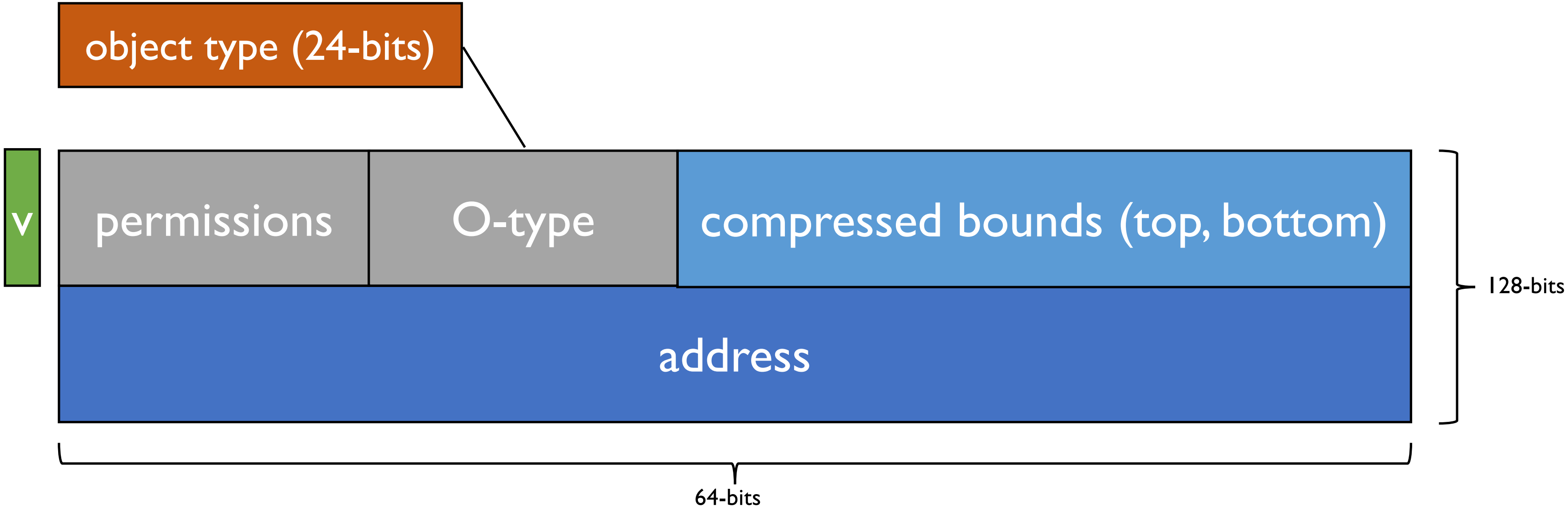
- Memory access
  - Loads and stores via a bounds checked capability
  - Exception if address is out of range
- Guarded manipulation of capabilities
  - Decrease bounds
  - Decrease permissions
  - Adjust the address
  - Extract/test fields

monotonic decrease in rights guaranteed by formally verified hardware

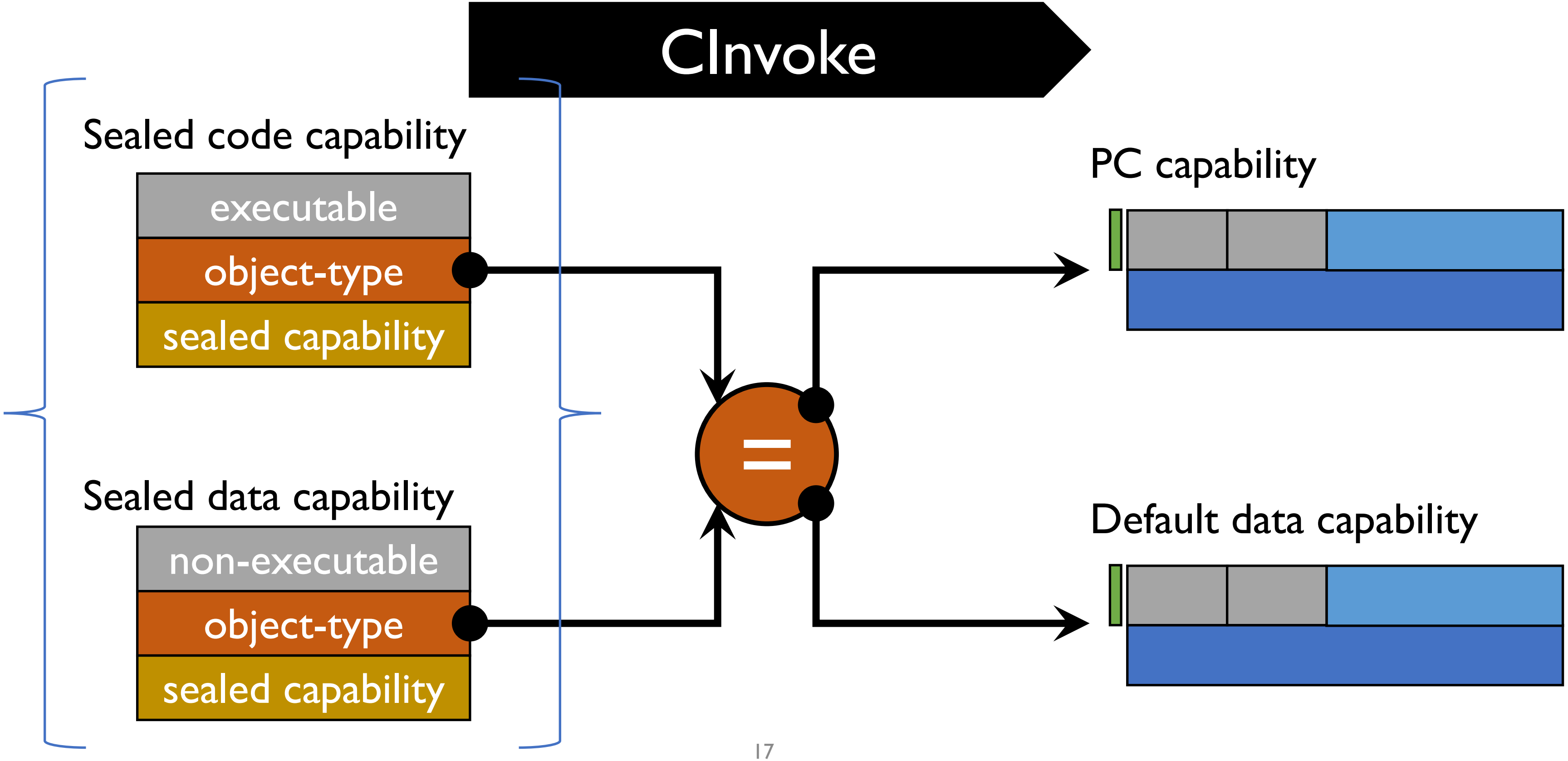
critical property for security

Sealed Capabilities for Compartmentalization

- Sealed capabilities are none dereferenceable capabilities
- Have to be unsealed (e.g. inside a compartment) before use



Calling a Compartment

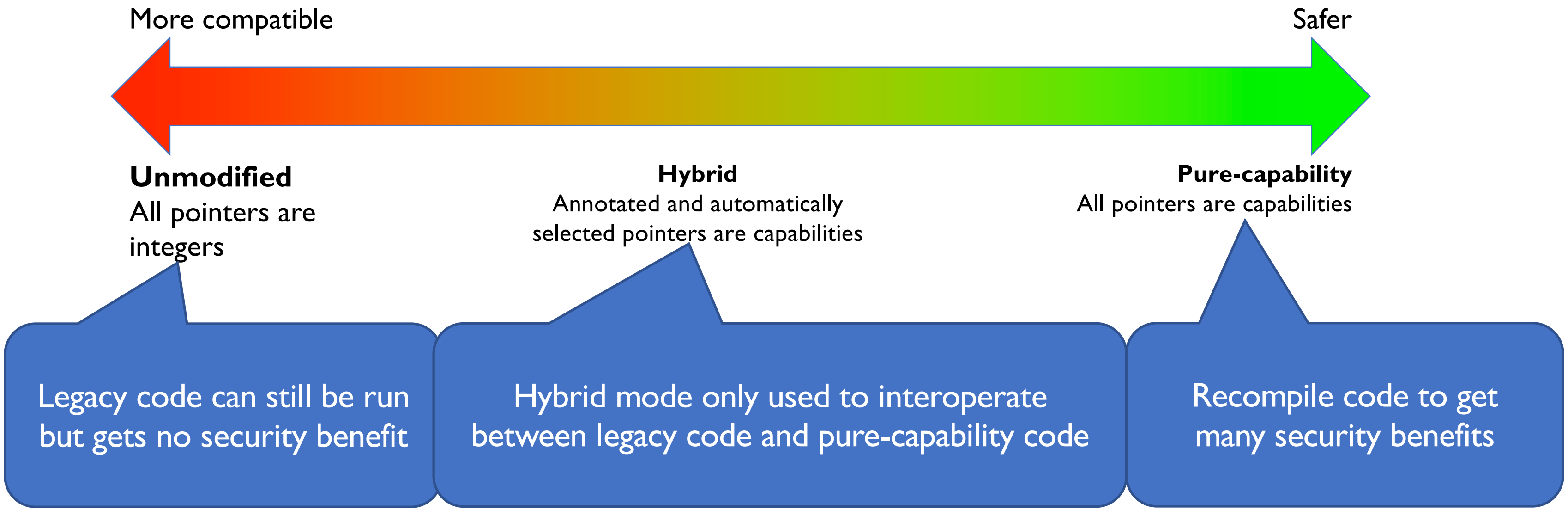


CHERI Software Models

Background to CHERI Software Models

- Machine-level capabilities and instructions provide the building blocks on which new abstract capability software models can be built
- Analogy:
  - Machine-level translation lookaside buffer (TLB) and page table walker enables the OS to represent virtual memory
  - Virtual memory can then be used in different ways to impose new security features, e.g. guard pages

Low-level CHERI software models



More compatible

Safer

Unmodified
All pointers are integers

Hybrid
Annotated and automatically selected pointers are capabilities

Pure-capability
All pointers are capabilities

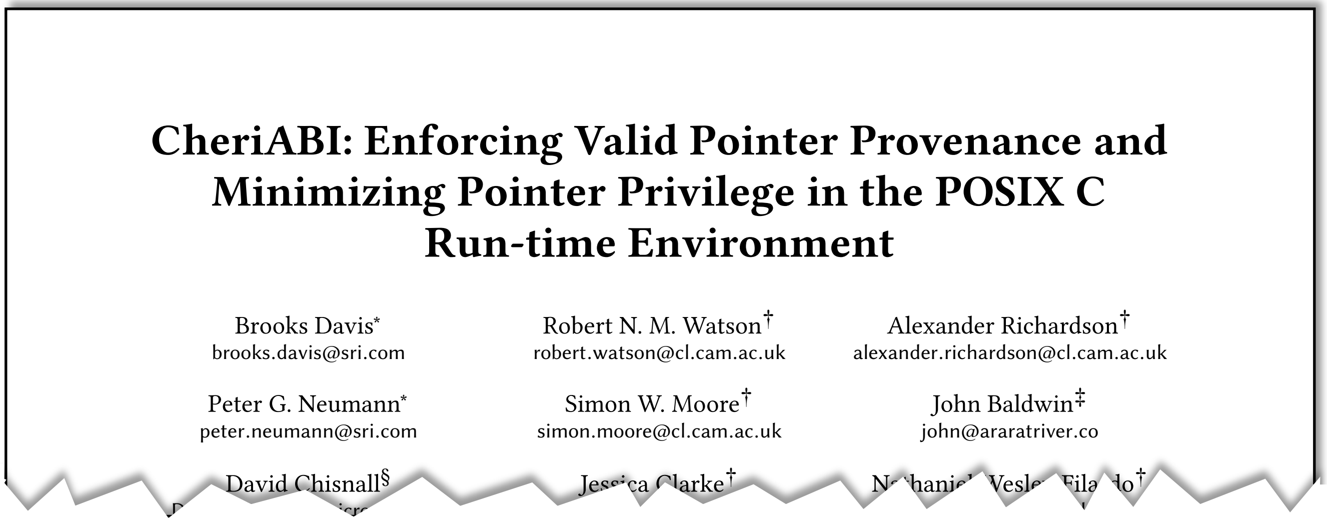
Legacy code can still be run but gets no security benefit

Hybrid mode only used to interoperate between legacy code and pure-capability code

Recompile code to get many security benefits

- Our **CHERI Clang/LLVM compiler** generates code for all three models

Pure Capability Code → Needs CheriABI



CheriABI: Enforcing Valid Pointer Provenance and Minimizing Pointer Privilege in the POSIX C Run-time Environment

Brooks Davis[*]
brooks.davis@sri.com

Robert N. M. Watson[†]
robert.watson@cl.cam.ac.uk

Alexander Richardson[†]
alexander.richardson@cl.cam.ac.uk

Peter G. Neumann[*]
peter.neumann@sri.com

Simon W. Moore[†]
simon.moore@cl.cam.ac.uk

John Baldwin[‡]
john@araratriver.co

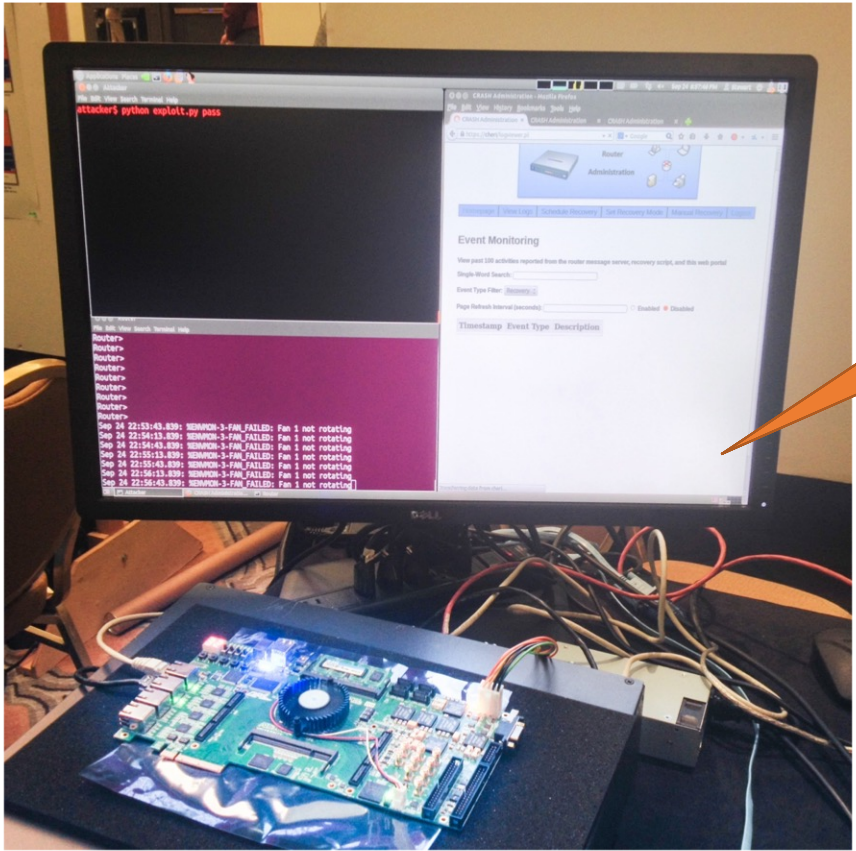
David Chisnall[§]

Jessica Clarke[†]

Nathaniel Wesley Filardo[†]

- Received best paper award at ASPLOS, April 2019
- Complete pure-capability UNIX OS userspace with spatial memory safety
  - Usable for daily development tasks
  - Almost vast majority of FreeBSD tests pass
  - Management interfaces (e.g. ioctl), debugging, etc., work
  - Large, real-world applications have been ported: PostgreSQL and WebKit

Red Team Evaluation by MIT Lincoln Labs



```
attacker$ python exploit.py pass
```

```
Router>
Router>
Router>
Router>
Router>
Router>
Router>
Router>
Router>
Sep 24 22:53:43.839: %ENVMON-3-FAN_FAILED: Fan 1 not rotating
Sep 24 22:54:13.839: %ENVMON-3-FAN_FAILED: Fan 1 not rotating
Sep 24 22:54:43.839: %ENVMON-3-FAN_FAILED: Fan 1 not rotating
Sep 24 22:55:13.839: %ENVMON-3-FAN_FAILED: Fan 1 not rotating
Sep 24 22:55:43.839: %ENVMON-3-FAN_FAILED: Fan 1 not rotating
Sep 24 22:56:13.839: %ENVMON-3-FAN_FAILED: Fan 1 not rotating
Sep 24 22:56:43.839: %ENVMON-3-FAN_FAILED: Fan 1 not rotating
```

Event Monitoring

View past 100 activities reported from the router message server, recovery script, and this web portal

Single-Word Search:

Event Type Filter: Recovery

Page Refresh Interval (seconds): ○ Enabled ● Disabled

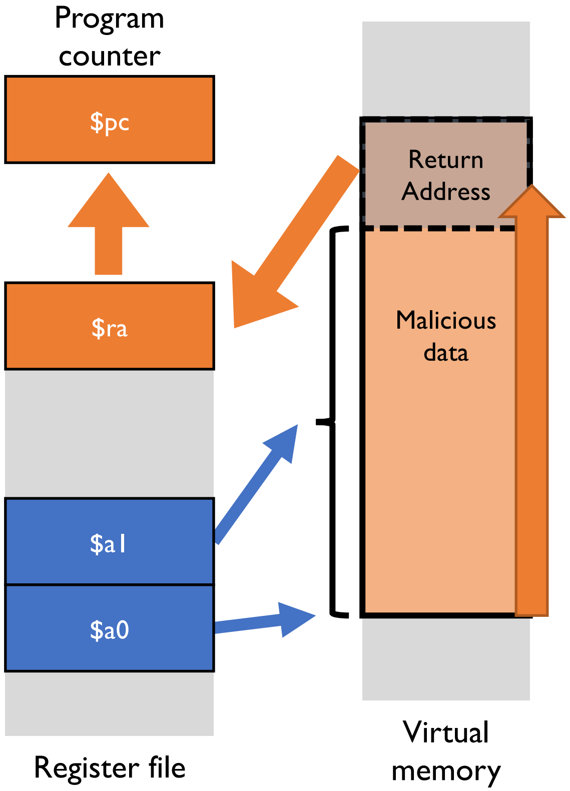
Timestamp Event Type Description

Attacker
File Edit View Search Terminal Help
Router
File Edit View Search Terminal Help
CRASH Administration - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CRASH Administration
CRASH Administration
CRASH Administration
https://cheri/logviewer.pl
Google
Router
Administration
Homepage
View Logs
Schedule Recovery
Set Recovery Mode
Manual Recovery
Logout
DELL

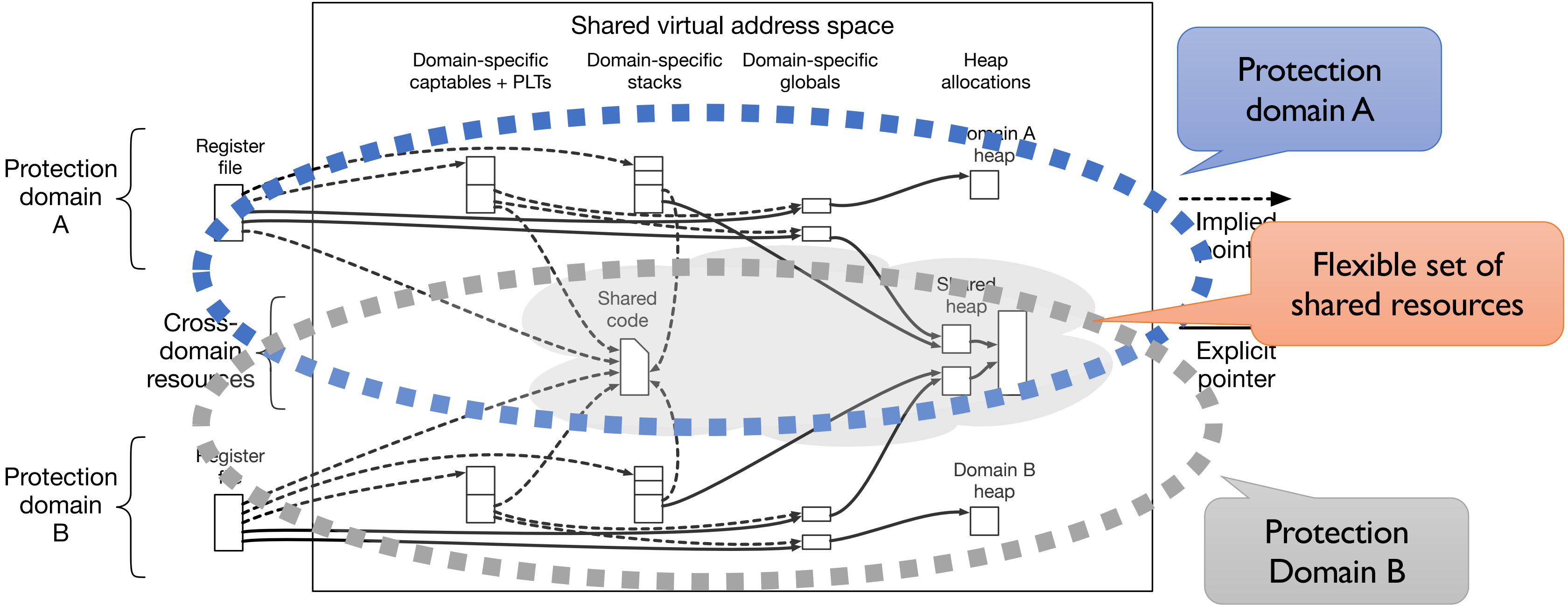
CHERI mitigates Heartbleed exploit!

Capabilities for Control-Flow Robustness

- Capabilities used for return addresses and other code pointers
- Integrity bit mitigates code reuse attacks:
  - ROP – Return Oriented Programming
  - JOP – Jump Oriented Programming
- Much better than current probabilistic technique ASLR (Address Space Layout Randomisation)



Example of CHERI-based compartmentalization

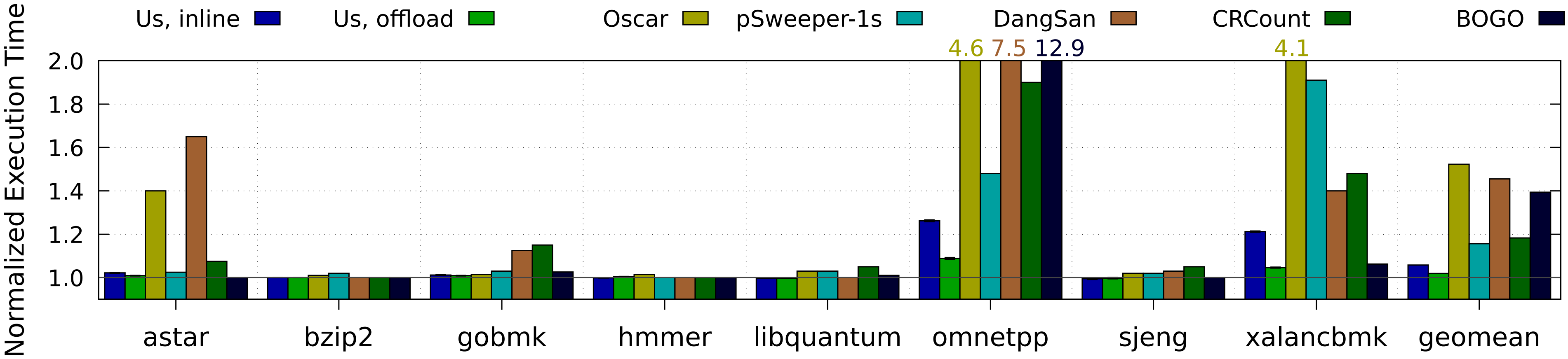


- Isolated compartments can be created using closed graphs of capabilities, combined with a constrained non-monotonic domain-transition mechanism

Temporal Memory Safety

- E.g. use-after-free vulnerabilities are common
- CHERI hardware directly implements strong spatial safety and enables efficient temporal safety:
    - Tags and object bounds held by capabilities makes software-based temporal safety efficient via revocation
    - Hardware optimisations makes software scanning for tags (i.e. scanning for object references) efficient

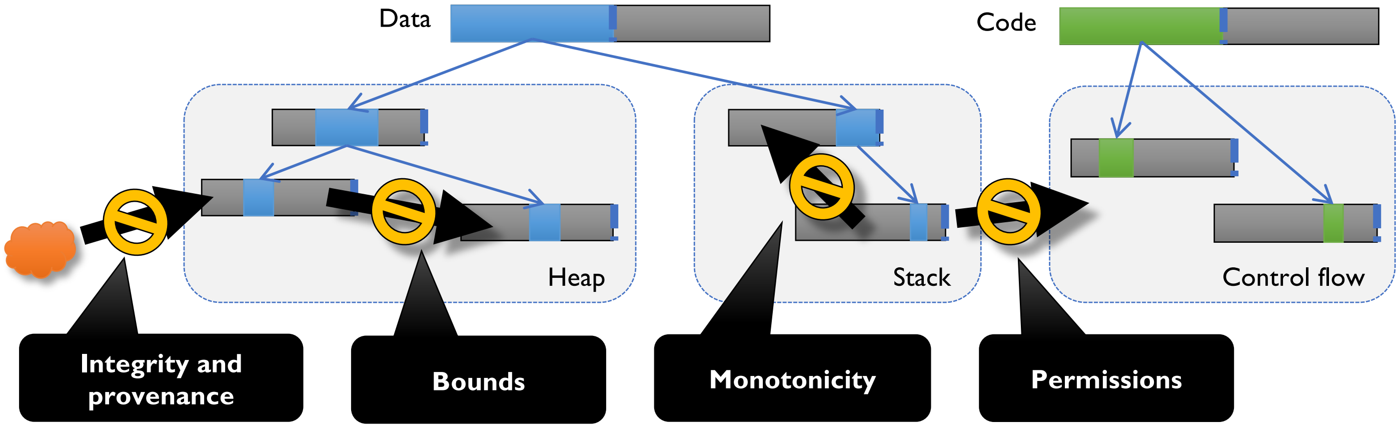
Cornucopia: State-of-the-Art Runtime Overheads



- CHERI-compatible SPEC CPU2006 benchmarks
  - *Without* second thread, geomean is 5.8% (worst 26.2%)
  - *With* second thread and core, geomean 1.9% (worst 8.9%)
  - When offloading, pause times 10-20% of single-thread sweep typical
- Applications unmodified beyond existing small patches for CheriABI compatibility

Research is ongoing to further optimize temporal safety!

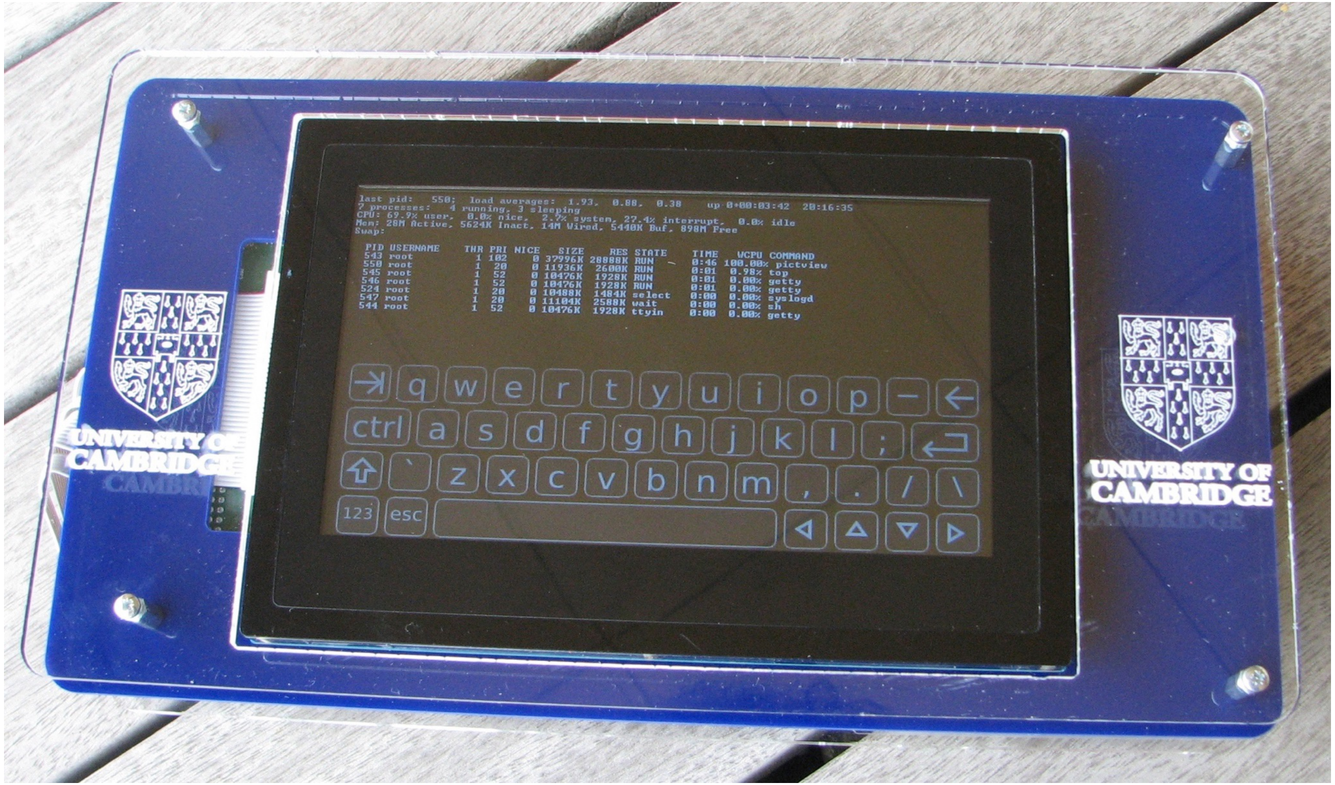
Summary of Capability Protections



- **Low-level capability hardware is a foundation for software models**
  - Guarantee a valid userspace pointer set with pointer privilege reduction
  - Highly efficient compartmentalization
  - Efficient, deterministic temporal safety

Building CHERI-RISC-V

First we made an FPGA-based hardware tablet



```
last pid:   550;  load averages:  1.93,  0.88,  0.38    up 0+00:03:42  20:16:35
7 processes:   4 running, 3 sleeping
CPU: 69.9% user,  0.0% nice,  2.7% system, 27.4% interrupt,  0.0% idle
Mem: 28M Active, 5624K Inact, 14M Wired, 5440K Buf, 898M Free
Swap:

  PID USERNAME     THR PRI NICE   SIZE    RES STATE    TIME   WCPU COMMAND
  543 root           1 102    0 37996K 28888K RUN      0:46 100.00% pictview
  550 root           1  20    0 11936K  2600K RUN      0:01  0.98% top
  545 root           1  52    0 10476K  1928K RUN      0:01  0.00% getty
  546 root           1  52    0 10476K  1928K RUN      0:01  0.00% getty
  524 root           1  20    0 10488K  1484K select   0:00  0.00% getty
  547 root           1  20    0 11104K  2588K wait     0:00  0.00% syslogd
  544 root           1  52    0 10476K  1928K ttyin    0:00  0.00% getty
```

Several Processors Implemented

- Early CHERI-MIPS: https://github.com/CTSRD-CHERI/cheri-cpu
- Current CHERI-RISC-V cores:
  - Piccolo 32b microcontroller: https://github.com/CTSRD-CHERI/Piccolo
  - Flute 64b/32b scalar core: https://github.com/CTSRD-CHERI/Flute
  - Toooba 64b out-of-order core based on MIT Riscy-OOO core https://github.com/CTSRD-CHERI/Toooba

Specification and Test

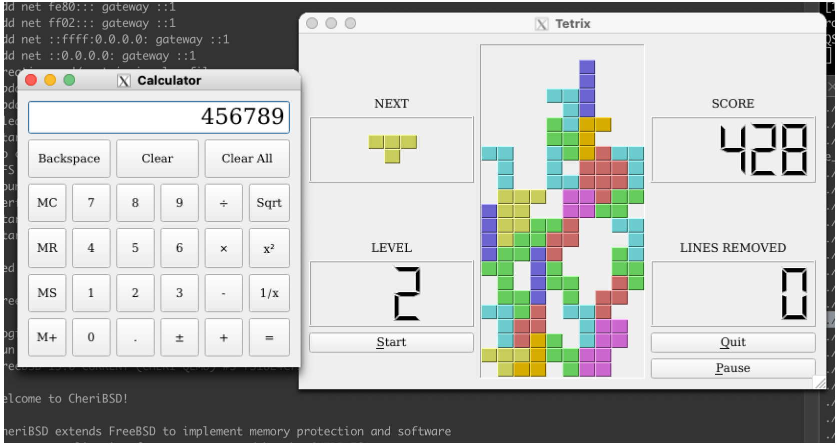
- SAIL-based executable formal model of RISC-V
  https://github.com/riscv/sail-riscv
  - Originally work from Cambridge but now the official RISC-V formal specification
- SAIL-based CHERI-RISC-V spec:
  https://github.com/CTSRD-CHERI/sail-cheri-riscv
- TestRIG for directed-random testing with test case shrinkage
  https://github.com/CTSRD-CHERI/TestRIG

Toolchain and OS support

- **C compiler** (Clang/LLVM) supporting capabilities
- **Full OS** (FreeBSD, FreeRTOS) that use capabilities for all explicit or implied userspace pointers
- Observations:
  - **Little or no software modification** (BSD base system + utilities)
  - Small changes to source files for 34 of 824 programs, 28 of 130 libraries
  - Overall: modified ~200 of ~20,000 user-space C files/header
    - Mainly localized to low-level run-time support

User space and demo applications

- Complete memory- and pointer-safe FreeBSD C/C++ userspace
  - **System libraries**: crt/csu, libc, zlib, libxml, libssl, …
  - **System tools and daemons**: echo, sh, ls, openssl, ssh, sshd, …
  - **Applications**: PostgreSQL, nginx, WebKit (C++)
  - **GUI:** X11 client libraries, Qt…



```
dd net fe80::: gateway ::1
dd net ff02::: gateway ::1
dd net ::ffff:0.0.0.0: gateway ::1
dd net ::0.0.0.0: gateway ::1

elcome to CheriBSD!

heriBSD extends FreeBSD to implement memory protection and software
```

Current Research Directions

- Compartmentalisation
  - Demonstrably much more efficient than process-based compartmentalisation
  - But need new software models
  - Need to ensure that CHERI compartments are robust even against transient executions attacks (see next lecture)
- Temporal memory safety
  - Microarchitectural and run-time optimisations
- CHERI for the whole SoC
  - CHERI for accelerators
- Toward exascale: CHERI for partitioned global address spaces
- CHERI everywhere: CHERI for x86
- Refining CHERI and the Morello architecture to bring it into the main stream Arm v9 ISA

Conclusions

- CHERI Provides the hardware with more semantic knowledge of what the programmer intended
  - Toward the **principle of intentionality**
- Efficient **pointer integrity** and **bounds checking**
  - Eliminates buffer overflow/over-read attacks (finally!)
- Provide scalable, efficient compartmentalisation
  - Allows the **principle of least privilege** to be exploited to **mitigate known and unknown attacks**
  - Large performance improvement over process-based compartmentalisation
- **Working with industry to bring the technology to market**
- Thanks to sponsors: DARPA, ARM, Google, EPSRC, HEIF, Isaac Newton Trust, Thales E-Security, HP Labs, Huawei

https://www.cl.cam.ac.uk/research/security/ctsrd/

Simon.Moore@cl.cam.ac.uk
Computer Science & Technology

Reading

- *The Arm Morello Evaluation Platform—Validating CHERI-Based Security in a High-Performance System* IEEE Micro 2023. DOI: 10.1109/MM.2023.3264676
  https://ieeexplore.ieee.org/document/9895591
- *Efficient Tagged Memory*, ICCD 2017
  https://www.cl.cam.ac.uk/research/security/ctsrd/pdfs/201711-iccd2017-efficient-tags.pdf
- *CHERIvoke: Characterising Pointer Revocation using CHERI Capabilities for Temporal Memory Safety*, MICRO 2019
  https://www.cl.cam.ac.uk/research/security/ctsrd/pdfs/201910micro-cheri-temporal-safety.pdf
- *Other papers/reports:*
  - Companion to CHERIvoke: *Cornucopia:Temporal Safety for CHERI Heaps*. In Proceedings of the 41st IEEE Symposium on Security and Privacy (Oakland 2020). San Jose, CA, USA, May 18-20, 2020
    https://www.cl.cam.ac.uk/research/security/ctsrd/pdfs/2020oakland-cornucopia.pdf
  - For reference: Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 9)
    https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-987.html
  - More microarchitecture: *CHERI Concentrate: Practical Compressed Capabilities*, IEEE Transactions on Computers 2019
    https://www.cl.cam.ac.uk/research/security/ctsrd/pdfs/2019tc-cheri-concentrate.pdf