

## CHERI technology overview

A quick introduction

### **Executive Summary**



#### Solving the worst cybersecurity issue

- Cyberattacks cause more than \$10 trillion of damage / year
- Most attacks due to a lack of Memory safety
- Unsolvable problem with traditional software solutions (new languages, practices, tools...)
- CHERI technology solves the memory safety problem



Source: Trends, challenge, and shifts in software vulnerability mitigation - Microsoft 2019

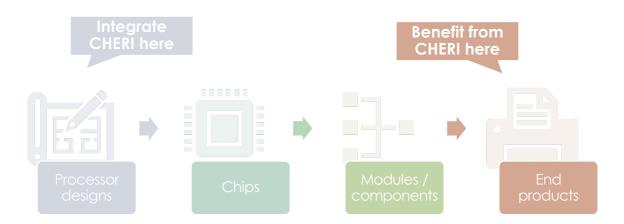


#### Hardware-based solution to a software problem

- CHERI
  - Open technology
  - Hardware protection at processor-level
  - Brings strong security to existing code
- Now out of the lab
  - 15 years of development by University of Cambridge (UK) / SRI (USA)
  - Formally proven
  - Demo and prototypes available now getting into products
  - Ready for industry to adopt



#### CHERI requires new hardware



#### CHERI backed by strong supporters























~ \$300 million investment in the development of CHERI

(by governments and industry)







#### CHERI benefits

Brings security even to existing code

No need to rewrite mostly recompile & optimize

Built-in compartmentalization secrets remain secret

Fast learning curve
 no need to change language, habits...

Low impact

Area & Power
 5% more at CPU level

Performance
 ~3-5% less (work in progress – still improving)

- Possible to increase performance with optimization
  - Use native capabilities instead of inefficient security software
  - Some case studies show up to 800% improvement!



# The memory safety problem

#### Data breaches are very costly

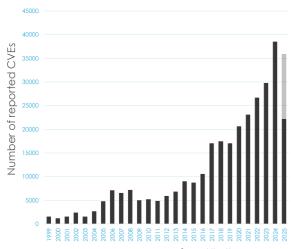
>\$500M

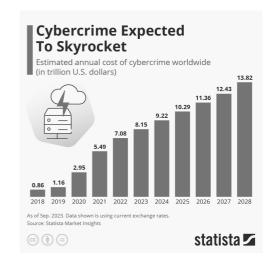
Cost of addressing Heartbleed buffer overflow vulnerability 6X

Increase in firmware attacks reported by NIST between 2017 and 2024 ~\$10T

Worldwide cost estimate of cyberattacks per year (and growing fast)

#### Software vulnerabilities are causing increasing risk





Source: https://www.cve.org



#### The main problem is (the lack of) memory safety

Memory abuse (e.g. buffer overflows) is the main attack vector

Constant ratio of over the past 20 years

• ... even with all the work done on software to avoid this!

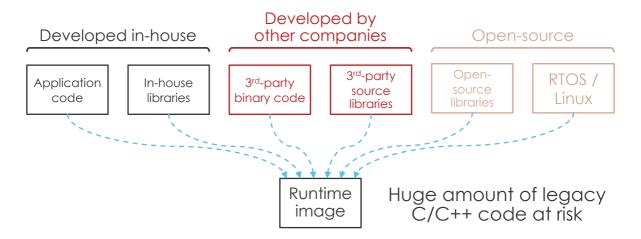


Source: Trends, challenge, and shifts in software vulnerability mitigation - Microsoft 2019





#### Impossible to re-write software to fix the problem



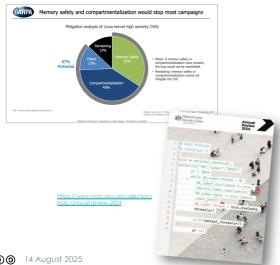


#### Possible solutions for memory safety

- Use "memory safe" languages like Rust or .Net
  - Requires rewriting trillions of lines of C/C++ code
  - Possible for new code, but no compartmentalisation
- Use "coarse-grained" techniques like stack "canaries" to detect issues
  - Helpful, but they statistically leave too many holes
  - Hacking techniques already developed
- Use "fine-grained" techniques like CHERI
  - Best option, but needs new hardware



#### Memory safety becomes a key topic







Article; https://bidenwhitehouse.archives.gov/opcd/briefing-room/2024/02/26/press-release-technical-report/





### CHERI technology

Capability

Hardware

**E** nhanced

**R**ISC

I nstructions



#### **About CHERI**



 Initiated by a project from



 Originally developed by



- Matured for 15 years
- Supported by











~ \$300 million investment in the development of CHERI

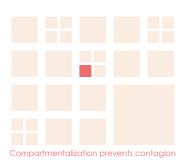
(by governments and industry)

#### ○ CHERI

A new hardware-based approach to memory safety

- Strong, fine-grained **memory protection** 
  - Hardware enforced
  - Deterministic

- Compartmentalization
  - Principle of least privilege



#### Main memory issues with C/C++

- Possible to access memory out of expected buffers
  - Access confidential data
  - Modify / delete critical data or code
  - Inject malware code
  - Spy on communications
  - Erase traces of attack
- Functions cannot protect their data from each other
  - Only works when the software can be trusted
  - Enable privilege escalation

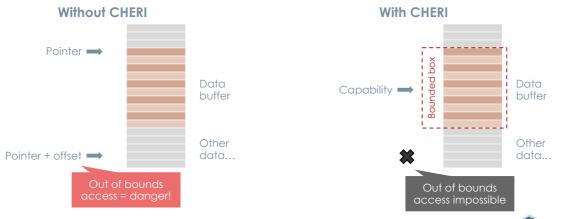
Need memory safety

Need compartmentalization



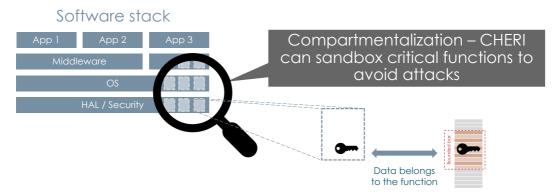
#### CHERI brings memory safety

Replacing pointers by capabilities – with hardware control



#### CHERI brings compartmentalization

 Capabilities belong to an identified function / execution context



#### CHERI relies on hardware protection

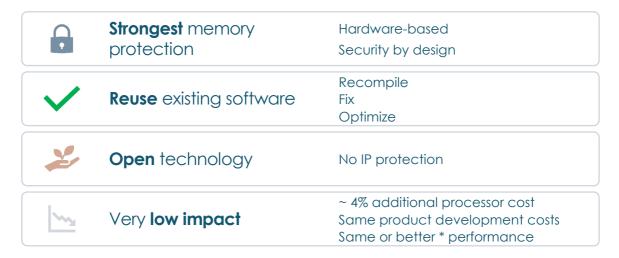
- CHERI requires adapted processor
  - Can be applied to any type of core
- Hardware-enforced security
  - Impossible to bypass by software
  - Formally proven
- Reuse existing code
  - Just recompile application
  - Choose which part to protect
- Benefit from CHERI
  - Rejects dangerous code
  - Create CHERI compartments for critical code



## CHERI adoption (costs, benefits)



#### Benefits of CHERI







#### CHERI has a positive impact

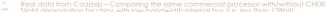
#### Limited costs

- Area impact
  - Processor only 4% larger\*
  - Similar power consumption
  - Similar performance\*\*
- Memory impact
  - Small area for tag storage
  - Double size for pointers (mostly impacts stack)
  - No change on data storage requirements
- Software development
  - Need adapted tools (open-source available)
  - Less than 0.5% application code\*\*\* to adapt
  - Need recompilation
  - OS / low-level drivers need work (done once)

CHERI Alliance helps by stimulating the ecosystem to adapt OS / software / tools

#### Huge gains

- Memory safety!
  - Save in patching costs
  - Compiler detects mistakes in existing code
- Performance gains\*\*\*\*
  - Remove or simplify software-based mechanisms (TEE, compartmentalization, security modes, ...)
  - Eliminate context switching in hypervisor
  - Reduce code, improve execution speed
- Fast, low-risk integration of unsafe code
- Save security experts' time
  - Not wasting time on bug hunting...
  - Reduce time of memory safety code review





<sup>\*\*</sup> Requires OS / security mechanisms adaptation (done once, usually by the ecosystem





#### O What makes CHERI different?

	Previous solutions	Problem	CHERI
Security control	Statistical  "likely" to detect a problem	Targeted attacks bypass protection Some problems detected too late	Systematic • 100% coverage
Enforcement	Some complex, disparate hardware features, no coherency across architectures Rely on "trusted" software and/or explicit checks	Additional complexity Software can be hacked (and has been)	Hardware-enforced     Simple, holistic protection     No way to bypass by software (unforgeable tags)
Software impact	High impact on software  A lot of additional code needed to protect and isolate  Need best security experts to review all software stack  Often need to rewrite code	Difficult to catch all issues Reduce performance and increase code size Experts are scarce	Need recompilation     Nead recompilation     Adapt some very low-level code     Fix previously undetected problems
Type of solution	Reactive • Fix problem if vulnerability discovered Proactive solution possible (but uneconomical)	Huge exploitable attack surface, susceptible to 0-day attacks Most systems are not upgraded immediately / regularly	Preventive Protects against existing and to- be-designed attacks on memor





#### Easy software migration

- CHERI-enabled processors can run legacy code \*
  - "Hybrid mode" run CHERI and non-CHERI code simultaneously
- Recompile / adapt / optimize code progressively
- Major OS are being ported to CHERI
  - Driven / coordinated by the CHERI Alliance
  - FreeRTOS, Zephyr, Linux, FreeBSD, Linux, seL4, ...





#### Easy software migration – Typical paths

Typical bare-metal / RTOS development



Typical rich-OS development







e.g., user-racing arrays – skip this stage for an kilos as many low-level functions are abstracted Optional: may require some re-architecting of code

#### Benefits of CHERI for functional safety

- Reduce development and certification costs
  - Detect programming mistakes faster
  - Simplify the analysis of source code
  - Demonstrate absence of memory vulnerabilities
- Allow modern development
  - No need for strict MISRA-C or SAFE-C constraints
  - Use all features of the C language
  - Enable innovation and new features
- Enable consolidation of multiple functions on a chip
  - Watertight isolation
  - Reduces cost and complexity



#### CHERI backed by strong supporters





















many other members of the CHERI Alliance <a href="https://cheri-alliance.org/member/">https://cheri-alliance.org/member/</a>

#### CHERI projects / products

- Prototypes / proof of concept
  - Proof of concept
    - arm Morello Program
  - Open-source / prototype
    - UNIVERSITY OF CAMBRIDGE Microsoft
- Commercial use
  - Already available from

    - Codasip lowRISC



- CHERI RISC-V standard ratified soon RISC-V°
- OS ported to CHERI (Free RTOS, Zephyr, Linux, FreeBSD, seL4...)

Now collaborating as part of the

#### **CHERI Alliance**

- Support HW/SW ecosystem
- Solve common problems
- Joint promotion



## 99

The CHERI architecture's support for fine-grain memory protection and scalable compartmentalization promises to revolutionise our ability to protect personal data and provide strong defences against malware on mobile devices and in the cloud

Ben Laurie, Director of Security,
Google Research





79

As noted by the White House in a recent report on a path toward secure and measurable software, hardware support is critical to robust and efficient memory safety. Compiling software to run on CHERI enhanced processors guarantees very strong memory safety that an attacker cannot bypass

Professor Simon Moore, **University of Cambridge** 





Contact contact@cheri-alliance.org

Web www.cheri-alliance.org

