



AdaCore CHERI Cyber Security Evaluation Summary Report

Abstract: This report summarises the investigations and results of a cyber security study into the Capability Hardware Enhanced RISC Instructions (CHERI) microprocessor architecture. The security evaluation described in this report was performed as part of the Edge Avionics project ('Edge Avionics'), funded by the Rapid Capabilities Office (RCO) of the UK Royal Air Force (UK RAF).

Introduction

During the Edge Avionics program, AdaCore evaluated security claims made by the CHERI community through the Digital Security by Design (DSbD) initiative. The goal was to inform Edge Avionics project partners about the level of security assurance offered by the Arm Morello CHERI architecture via evidence-based arguments made by AdaCore cyber security experts.

The investigation strategies, security verification specifics, and results are publicly available. More specifically, during the Embedded Real-Time Systems conference in Toulouse, France, in 2024, AdaCore presented a paper titled "Security by Default - CHERI ISA Extensions Coupled with a Security-Enhanced Ada Runtime." The paper, which has since been published in the conference journal <ERTS 2024 Tu.4.c.1>, discusses security claims around CHERI and the direct role the hardware can play in achieving security airworthiness. The paper can be read via our tech papers website:

<https://www.adacore.com/papers/security-by-default-cheri-isa-extensions-coupled-with-a-security-enhanced-ada-runtime>.

Furthermore, all materials and techniques required to reproduce the security verification tests are publicly available via AdaCore on GitHub: <https://github.com/AdaCore/cheri-examples>.

In addition to the findings summarised below, AdaCore found two faults within their Ada runtime software through testing on a CHERI architecture. This report will provide a high-level overview of the CHERI-discovered defects. Without CHERI, the resultant state from unsafe memory calls would have been challenging to identify, diagnose, and determine the underlying fault. Failure to identify software faults could have led to the release and deployment of vulnerable software. Our ERTS paper explores the verification benefits of Morello (in addition to the multiple deployment security benefits), particularly around aviation



cyber security and safety certification as described in guidelines like DO-178C / ED-12C and DO-326A / ED-202A.

Beyond Language-defined Runtime Checks

CHERI is language agnostic. Assuming the memory allocation facilities of the deployed Operating System, the language runtime and compiler toolchain fully utilize the CHERI hardware components (more specifically, ensuring CHERI "pure capability mode" and bounding memory allocations to support the principle of least privilege), CHERI will extend memory safety checks beyond what is possible with memory-safe programming languages (like Ada, SPARK, and Rust). CHERI should be seen as a security toolkit that needs to be utilized appropriately by the executing runtime software; we cannot claim elevated security assurance from the deployment of CHERI hardware when software memory allocators fail to correctly use the CHERI hardware protection mechanism such that out-of-bounds memory accesses to adjacent allocations are not guarded. Verification is required to ensure software-driven memory allocators correctly use the CHERI hardware security feature set. AdaCore's GNAT Pro for CHERI toolchains contain CHERI pure capability memory allocators, and all memory allocations use capabilities at the finest grain compartmentalization possible. The specifics behind CHERI security mechanisms and how they expand beyond programming language runtime constraint checks are explained within our ERTS paper. However, the following two software bugs are evidence of this behavior as they are only identifiable when executing tests on a CHERI microprocessor architecture.

Secondary Stack bug

While adapting the GNAT Pro Ada compilers for Morello, we uncovered a previously undetected issue in a development version of the compiler. This issue caused the compiler to incorrectly allocate memory, leading to the potential for out-of-bounds memory access.

The problem arose in a specific hard-to-detect corner case scenario: when a function returned a dynamically sized object (one whose size is only known when the program runs), the compiler allocated too little memory to store it. This mistake meant the program could unintentionally access memory beyond its allocated limits, possibly interfering with other data and creating a security risk.

This issue went unnoticed in standard testing environments because conventional systems could not identify such an anomaly. However, when we ran our test suite on CHERI using its advanced memory safety features, the problem became immediately evident. Even specialized tools like Valgrind and AddressSanitizer failed to detect the vulnerability.

By catching this issue early, we prevented a potential vulnerability from reaching production (GNAT Pro Ada runtimes are present on many defense platforms).

Environment Task Stack Size Bug

The technical specifics of this bug are similar in complexity to the above and act as another example of how CHERI catches obscure corner-case software defects that can lead to security vulnerabilities that can be exploited for malicious intent. While modifying the minimum size of the task-specific stacks in our tasking runtime, we missed a corner case in the special handling of the environment task. By verifying our software on CHERI, the problem was identified straight away. Without CHERI, the bug would have likely ended up in the released compiler and deployed into operating platforms.

Testing CHERI Security Claims

AdaCore used several exploitation techniques to counterclaim the cyber security benefits of CHERI microprocessor architectures. In all cases, the resultant state of a system following a successful attack will be indeterministic such that the satisfaction of functional requirements can no longer be guaranteed; if an attack is successful on a deployed platform, the impact will range from the unavailability of mission-critical functionality to critical failure leading to a safety hazard.

Typical (traditional) attacks require the delivery of payloads into the deployed system. More advanced techniques involve manipulating the program state to perform the attacker's desired behavior. In addition, modern-day attack techniques commonly use malicious actors to deploy "supply chain attacks."

AdaCore investigated and developed examples of these attack patterns against systems executing on CHERI architectures. In all cases, AdaCore used an experiment control first to demonstrate that an attack worked on a non-CHERI comparable architecture; we tested on Aarch64 and Arm Morello architectures.

Traditional Memory Corruption Attack

A typical example is forcing a buffer overrun and writing a jump address into memory to execute malicious code instead of the desired code. AdaCore verified that CHERI can prevent execution from undesirable memory regions. ([ERTS Paper](#)).

Return Oriented Programming (ROP) Attack

ROP is a technique for exploiting a software bug for malicious intent. More specifically, this technique involves circumventing memory protection techniques that protect program code from modification. Traditional exploitation in ROP attacks consists of a hacker modifying the underlying functional behavior of the system to perform undesirable behavior. AdaCore concluded that such an attack is significantly challenging to perform on CHERI, and should an

attack be possible, ChERI compartmentalization mechanics will vastly reduce the impact. ([ERTS Paper](#) / [GitHub Example](#))

Supply Chain Attack

AdaCore verified that a supply chain attack *could* be executed through malicious third-party libraries on a ChERI architecture when the software runtime does not fully utilize a library-level compartmentalization security measure. AdaCore validated that CheriBSD compartmentalization works as intended on shared libraries. More specifically, we created a malicious shared library that, without compartmentalization, could modify the main program's state. With compartmentalization, ChERI blocked the attack, and we could not circumvent these restrictions; after enabling the security mechanism, AdaCore could not devise a secondary attack path. ([GitHub Example](#))


Logic Error Attack

ChERI does not provide memory safety over logic errors. It does not understand the functional requirements of the executing systems. Traditional and well-established high-integrity software development verification techniques should still be used to argue that an application is functionally correct.

Security Implications of ChERI Limitations

The Arm Morello realization of the ChERI microprocessor architecture contains a few notable limitations that undermine the DSbD security principle of least privilege. More specifically, we can categorize these limitations as performance-related hardware design decisions over compressed capability bounds formats and a lack of fine-grained memory allocation for composite-type components. AdaCore didn't explicitly investigate the security implications of these limitations as, in both cases, the runtime constraint checking feature of the Ada programming language is considered sufficient to guard against unsafe memory usage.

The first issue is that the Morello architecture uses a compressed bounds format, ensuring the bounds are precise for objects up to 4 KB. For objects larger than 4 KB, however, the bounds must be aligned to increasingly more significant powers of 2 address boundaries. The compressed format prevents the bounds for large objects from being represented precisely and thus requires padding in the memory allocation to align the capability bounds, meaning that ChERI will not detect minor accesses past the end of the array in the padding area. We consider the security implications of exploiting a memory read into a padding field to be low; however, it is also not considered an issue for GNAT Pro Ada on ChERI as the semantics required by Ada's language-defined checks ensure the raising of a `Constraint_Error` exception before accessing an array with an index value that is not within the bounds of the array index type. The rules of the Ada programming language enforce



precise bounds checking, even for large bounds. While CHERI's bounds checking is also precise in most cases, there is room for error with large data structures (objects larger than 4 KB).

The second issue is limited capability memory allocation with C Structs and Ada Records. More specifically, there are scenarios where C Struct Members or Ada Record Components could overflow and corrupt a neighbor without CHERI identifying or guarding against the unsafe memory access (a capability is only created for the composite type - not the field elements). AdaCore didn't include an assessment of the security implications of this limitation as it's only an issue for C Structs; any Ada record component defined as an array type (and therefore susceptible to an overflow) is already bounds checked by the GNAT Pro for Ada runtime (i.e., if you try to access an element outside of the bounds of the array, a run-time Constraint_Error is raised).

Therefore, CHERI (or at least with the Arm Morello implementation) is not enough to guard against all unsafe memory access scenarios with applications developed via a "memory-unsafe" programming language. While there are still clear benefits in executing C code on CHERI, it's only by combining a memory-safe programming language like Ada with memory-safe hardware like CHERI that we could begin to claim the highest level of security assurance.

Conclusion

Our security research and CHERI developer experience have shown that CHERI architectures provide two key benefits:

1. CHERI is an excellent memory-safe verification environment that AdaCore now uses to test software and elevate security assurance. We recommend others do the same.
2. CHERI provides hardware security mechanisms that, when fully utilized by the operating system, language runtime, and compiler (as is the case with AdaCore's GNAT Pro Ada runtimes), significantly reduce the risk of unauthorized and malicious access to the executing platform application (across multiple exploitation techniques).
3. The security benefits of CHERI go beyond the feature sets of all recognized memory-safe programming languages and provide security assurance over language aspects like Ada's "unchecked programming," Rust's "unsafe" mode, and user-specified assumptions within formal proof languages such as SPARK.
4. To achieve the highest level of security assurance, memory-safe hardware (CHERI) should be combined with memory-safe programming languages (Ada); the limitations of one are guarded against by the feature set of the other.