Secure memory management is a vital low-level component of a security infrastructure, providing another layer of protection for higher-layer vulnerabilities. This is constantly shown through critical CVEs, such as the recent Windows TCP/IP RCE vulnerability exploiting an insecure handling of IPv6. Capability Hardware Enhanced RISC Instructions (CHERI) is a specification of extensions to standard ISAs that provides precise memory protection and compartmentalisation to mitigate memory vulnerabilities such as a buffer overflow attack, which was what exploited Windows' insecure IPv6 handling. CHERI also defines capabilities: secure pointers that contain an address and metadata including bounds, permissions and a validity tag to enforce fine-grained control of memory access rights. Started in 2010 as a collaborative effort by the University of Cambridge, CHERI primarily supports the RISC-V ISA, although an ARM Morello board was released in early 2022 which is CHERI-enabled.

Research done at the University of Glasgow that executed benchmarks against popular CHERI allocators for the Cheri-BSD port has identified vulnerabilities and unexpected behaviour, leading to security flaws such as allowing access to previous data from old capabilities. I propose to develop a CHERI allocator in C that defends or mitigates these vulnerabilities, whilst not introducing any new weaknesses and minimising the effect on performance. Artifacts produced from development would include source code and documentation, shared and static libraries to link with C applications using the custom allocator, and potentially a build script to configure the environment and dependencies, as well as compile and link the allocator. I would also be interested in researching the feasibility of using the Cheri-Rust extension of Rust to develop a CHERI allocator, due to the language's increased memory and type safety. This aspect of the research would not likely yield any artifacts but would evaluate whether Cheri-Rust is suitable for creating allocators given the inherently unsafe nature of memory management, and if so whether it provides any significant benefits over languages such as C and C++.

This research would attempt to fix the vulnerabilities found during the benchmarks undertaken during the University of Glasgow's research on allocators described above, which has not yet been carried out. The motivation for this project is further justified since Cheri-Rust could create a safer allocator if it is possible to use, which would help build more robust CHERI implementations. Although there is currently a Rust compiler for CHERI being developed that forks from the existing Rust LLVM compiler, there are no allocators that are either written in Cheri-Rust or fully satisfy all the security benchmarks, making this research both unique and existing within a cutting-edge and important field of cyber-security.

Special support required for this project would consist of access to CHERI-enabled hardware for one or both of the RISC-V and ARM ISAs so that benchmarks can be run. This could be provided through either physical board copies or SSH login. The Terasic DE4 FPGA Board developed by Cambridge to implement and test CHERI-RISC-V would be my first choice, although a SiFive HiFive Unleashed board along with FPGA extensions could also be used. The ARM Morello board has CHERI extensions and is primarily used for research and development of the possible security benefits of CHERI on ARM. Since it is available to be purchased by universities for research purposes, it is also an ideal candidate to provide CHERI-enabled hardware.