Technology Access Programme

Digital Security by Design Technology Access Programme

Pytilia - cohort 2

Final Report

February, 2023

Digital Security by Design - final report

Please answer the following questions, detailing what you have worked on during the programme, the outcomes of your project, and any thoughts / feedback you have regarding the Morello & CHERI technologies. Please complete by 24th February 2023.

General Information

Name of person(s) writing the report	**Michael Allen**
Project Goal : what were you hoping to achieve?	*The aim of this project is to create a packet processing application, which receives incoming packets before classifying and dispatching them to one or more untrusted plugins for consumption.* *This application should be able to operate in two different modes:* • *The untrusted consumer plugin(s) should be run in the same address space as the packet processing application. CHERI capabilities should be used to send the data from the application to the plugin. The plugin should only be able to access the data within the specified capability bounds and should have correct permission enforcement.* • *The untrusted consumer plugin(s) should be run in separate processes to the packet processing application. A traditional IPC approach should be used to send complete packets to the plugin, for example sending packets over UDP sockets. The plugin should not use CHERI capabilities to ensure data security.* *Performance metrics can be recorded for both methods of operation, specifically packet processing latency and CPU utilisation. This allows for a fair comparison of these metrics between CHERI-based security and a traditional IPC approach.*

1. Please provide a high-level overview of your activities during the programme.
(Max 6 bullet points; max 600 words total)

- **DPDK Port** - The DPDK framework is a packet processing framework written in C. This library was ported to CHERI C, so that it could build in PureCap mode. The porting process mainly required changes to be made in config files relating to the Morello architecture which was not supported yet. One of the main problems encountered during this process was that the size of a void pointer was 16 bytes instead of 8, which was the expected number for a 64-bit architecture. Additionally, atomic operations were not supported. They were required for some of the DPDK drivers to build, but those drivers were not required for building DPDK as a whole. The completed DPDK port was limited to one process due to the missing contigmem and nic_uio kernel modules, which were not loaded successfully in this project.
- **Packet Stream Generation** - A Python script was created to generate packet streams and store then in a .pcap file. These streams contained varying packet counts and packet lengths. These files were used as input for the packet processing application
- **Packet Processing Application** - A packet processing application was created using the DPDK port. This application takes a specified packet stream as input, reading each of the packets in. The application loops through each packet, classifying it according to arbitrary criteria for this project (the first byte of the packet), and then dispatching the packet to the relevant consumer. This application can dispatch the whole packet through UDP sockets, the corresponding CHERI capability to a consumer in the same address space, or do no processing on the packet. The method of dispatch is determined by a relevant command line argument or environment variable. There are other arguments/variables to make the application run in a quiet or verbose mode. A bash script was also created to simplify executing this application.
- **Capability Validation** - Before a capability is sent to a consumer, in the CHERI mode of operation, the bounds are set to match the packet length and write permissions are removed. To demonstrate this works correctly, validation checks can be ran through the use of a command line argument or environment variable. To test the bounds, an attempt to read one character beyond the packet length is performed. To test the permissions, an attempt to write to the packet data is performed. Both tests show that the appropriate error is raised.
- **Listener Application** - A listener application was created to receive the packets sent when using UDP sockets in the packet processing application. This application can be built and run in PureCap mode without error. However, when recording the performance metrics described below, the application was built in hybrid mode with no capabilities

used for fairness.

- **Performance Metrics** - Python and bash scripts were created to record the average packet processing latency and CPU utilisation for the packet processing application in all three modes of operation. The mode which does not process the packet allows for the packet processing time to be calculated, ensuring a fairer comparison of results between the other two operation methods. Measurements were taken for a range of packet streams, varying packet count and packet length. These results were imported to a Python script which parsed them and plotted relevant graphs. This showed that CHERI capabilities provided performance benefits throughout, with greater benefits provided as packet size increased.

2. Please provide a brief description of any new libraries and/or C/C++ code you ported to CHERI C/C++, including its function(s), what compiler it is usually compiled with, and what OS it is normally run on.

DPDK v20.11.1 was ported to CHERI C on the Morello architecture. DPDK (Data Plane Development Kit) is an open-source library used for packet processing originally created by Intel. The supported operating systems are Linux, Windows, and FreeBSD. Both the GNU and Clang/LLVM toolchains are supported when building DPDK. DPDK contains a range of example applications and uses, however, for this project the required functionality is reading in packets from a .pcap file. DPDK v20.11.1 was previously ported to the Morello FVP, however this port was not compatible with the physical Morello architecture.

3. Please describe how large the application libraries/code you ported to CHERI C/C++ are, referencing lines of code in your software and/or lines of code in dependencies/packages where possible.

Git shows the following change statistics for DPDK v20.11.1, between the previous Morello FVP port and the current version:

- 31 files changed
- 49 insertions
- 148 deletions

The packet processing application was based on the previous application built by Pytilia, Limelight. Git diff shows the following information:

- 19 files changed
- 1129 insertions
- 393 deletions

The listener application was created from an empty repository, and it contains 267 lines of code.

4. Were there significant missing software dependencies outside of your own software package(s) that had to be adapted?
If so, please describe them.

No software dependencies outside of DPDK had to be adapted for this project.

5. How helpful were CHERI C/C++ compiler warnings and errors in software porting to CHERI C/C++?

The CHERI C compiler messages were useful for showing pointers which could not be dereferenced, allowing for appropriate fixes to be made.

6. What other benefits of porting software to CHERI did you notice?
Eg: reducing resource / time requirements; directives for improving code, etc.

The code generated was intrinsically secure. Specifically, there were multiple bounds errors in the code which were found during development, and fixing these errors made the code safe.

7. Did you face any difficulties when porting your software to CHERI C/C++, and if so, how did you resolve them?
Describe any specific issues you faced in getting your software to compile and run, in hybrid

and/or pure-capability mode, as appropriate to your work.
In hybrid mode: N/A. The only time hybrid compilation was used was for the listener application. This application was built to compile in PureCap initially, so it was a seamless transition. **In pure-capability mode:** Atomic operations seemed to be incompatible with CHERI C, which were required in some of the DPDK network drivers. More specifically, it was atomic compare and swap operations that did not work and caused the DPDK build to fail. This was resolved by removing the drivers from the build, as they were not required for DPDK to build. This is because they were only required for 32-bit architectures, whereas the Morello architecture is 64-bit.

8. In light of the above, would you consider the effort of porting and compiling to be a blocker in your considerations around future adoption/use of CheriBSD? Please provide your thoughts for porting/compiling for both hybrid and/or pure-capability mode, as appropriate to your work.
In hybrid mode: N/A. **In pure-capability mode:** The time and effort spent resolving issues faced during the CHERI porting and development process was relatively small compared to the whole time spent in the development process. Specific to this project, the performance improvements, especially with packet processing latency, were very good. Also, the capabilities provided sufficient security within the developed applications. Bearing all of this in mind, the security and performance benefits that CHERI brought far outweigh the cost of extra development work.

9. What techniques did you use to confirm that the software worked correctly on CHERI/Morello? Please describe the techniques used, as well as any difficulties in this regard.
For the CHERI-enabled plugin, the CHERI capability bounds and permissions were set

programmatically before dispatch.

A command line option or environment variable can be used to attempt to read beyond the capability bounds, which correctly raises a capability bounds error which raises an in-address space security exception. GDB reveals this to be a capability bounds fault, as expected.

Furthermore, the write permissions are removed from a packet before dispatch. A separate command line option/environment variable will attempt to overwrite a character within the packet after the consumer has received the capability. Similarly, this raises an in-address space security exception, which GDB reveals to be a capability permission fault as expected.

10. Do you consider your software porting efforts a success?
Please describe the results of your porting and whether you were able to get your software to compile and run to your satisfaction.

The DPDK port allowed for a packet processing application to be created as desired. The required functionality of this DPDK port was reading in packet streams, which was achieved.

However, the port of DPDK is not based on the most recent release, 22.11.1, but instead 20.11.1. After the packet processing application was made, a brief attempt was made to recreate the port on DPDK 22.11.1, however due to time constraints this has not been completed.

Furthermore, this port of DPDK can only use one process at a time. This is because FreeBSD requires the contigmem and nic_uio kernel modules, which are provided with DPDK, to be loaded. These modules were not ported to CheriBSD, therefore the command line arguments --no-huge and --no-shconf must be specified to run the application. This limitation does not affect the packet processing application, however.

11. Did CHERI help you identify any bugs or potentially exploitable vulnerabilities in your software?
If so, please describe the vulnerabilities uncovered and how they were highlighted.

Most programming errors that were found were agnostic to the fact that CHERI C was used instead of regular C. However, using CHERI capabilities also made it easy to spot errors which are exclusive to CHERI C in the first place. For example, in the packet processing application, the packet contents are stored in a buffer, which is a CHERI capability. Once the buffer was cleared to be reused, if a larger packet was stored in the same buffer, a capability bounds error would have been raised due to tightening the bounds according to the first packet stored there. This identified that the CHERI validation occurred on the wrong capability, meaning that the capability being dispatched to a consumer in the same address space was not validated.

12. Would using CHERI and Morello reduce your software update frequency?
If so, by how much, and would you be able to quantify this as a cost saving?

In principle it should, however we don't have a precisely comparable non-CHERI product to use as a baseline.

13. How did you make use of CHERI capabilities during this project?

CHERI capabilities were used to ensure the security of using an untrusted 3rd-party consumer plugin inside the same address space as the trusted packet processing application. This involved sending the capability, which points to the received packet data instead of the packet data itself, to the consumer. These capabilities had their bounds set to match the packet length and had their write permissions removed. It was from these capabilities that the packet consumers were able to interact with the packets they received. This is in contrast to the IPC approach taken, where the entire packet content was sent across UDP sockets to the consumer.

14. What techniques did you use to evaluate CHERI/Morello's ability to provide security enhancements and/or reduce external vulnerabilities in your project?

Once the CHERI-enabled plugin received the capability pointing to the packet stored in the packet processing application, it was able to interact with the packet. This interaction was

limited based on the capability bounds and permissions. The bounds were tightened to match the packet length before dispatch to the consumer. This means that no malicious or poorly written plugins could access any other data from the packet processing application outside of the bounds of the packet's capability. Additionally, the write permission was removed before dispatch to the consumer. This meant that the consumer could not modify the packet that was being stored within the packet processing application, but instead it could only read the contents of the packet.

15. Were you able to prove that CHERI/Morello was able to prevent any attacks or else add any additional levels of security to your application?
If yes, please share some high level results of the tests/evaluations mentioned above.

For the CHERI-enabled plugin, the CHERI capability bounds and permissions were set programmatically before dispatch.

A command line option or environment variable can be used to attempt to read beyond the capability bounds, which correctly raises a capability bounds error which raises an in-address space security exception. GDB reveals this to be a capability bounds fault, as expected.

Furthermore, the write permissions are removed from a packet before dispatch. A separate command line option/environment variable will attempt to overwrite a character within the packet after the consumer has received the capability. Similarly, this raises an in-address space security exception, which GDB reveals to be a capability permission fault as expected.

16. Did you notice any performance trade offs when using CHERI / the Morello board?
If so, please describe them below.

Sending secure CHERI capabilities resulted in a much shorter packet processing latency than sending the whole packet content through UDP sockets. The latency was almost unaffected by packet size when using CHERI capabilities, but showed a positive correlation when using IPC. This shows that the benefits of using capabilities increases as packet size increases.

Furthermore, the CPU utilisation decreased when using CHERI capabilities. The difference in

CPU utilisation also increased as packet size increased.

17. Please upload a video/screenshare demonstrating your achievements of your work during this programme.
Please copy/paste a link below.

https://www.youtube.com/playlist?list=PL4Qr-4Vy_oo1T-6BbNMtqZG5YBmmzJfdx

18. Please upload any additional documents that you feel demonstrate your achievements during this programme.
Please copy/paste a link to a folder/file below.

https://www.dropbox.com/s/ye02anqr7pp91k5/Project_Summary.pdf?dl=0

https://github.com/MichaelTA0111/DPDK-20.11.1-Morello
https://github.com/MichaelTA0111/CHERI-Networking
https://github.com/MichaelTA0111/Packet-Generator
https://github.com/MichaelTA0111/Result-Plotter

19. Overall, do you feel you have achieved the goals of your project plan? If not, what areas of work still remain?

Yes; the overall aim was to get a fair comparison of performance metrics when using CHERI capabilities for security compared to a traditional IPC approach. This was achieved through creating a packet processing application and collecting the required statistics through the relevant bash/Python scripts. Plotting the graphs revealed the expected results which were hoped for, namely that CHERI capabilities provide a secure form of packet dispatch to consumers on the same machine at a faster speed with lower CPU utilisation.

The DPDK port worked as required, but a more up-to-date version of this framework would have been preferable, especially if multiple processes were allowed to run in one DPDK

instance.

20. Please give any comments and observations on the usability of CHERI/Morello.

If someone is familiar with C, it should be fairly straightforward to start writing programs using CHERI C with a small learning curve. Based on the limited experience of this project, the porting process can be very tedious if the library is very big. However, writing new C code in the CHERI dialect much easier than the porting process, and had very little impact on the time taken to develop something new from scratch, assuming the required dependencies were already CHERI-compatible.

The Morello architecture is not very well supported when determining the build configurations, which is to be expected for a new release. As time progresses, this shouldn't be an issue as more libraries add in Morello-specific build configurations.

It is great to see that a desktop environment is provided with CheriBSD 22.12, which should make the OS much more usable in the future. Unfortunately, CheriBSD 22.05 was used throughout this project and, due to time constraints, there was no chance to upgrade to CheriBSD 22.12 to test this feature.

21. Are there any elements of Morello / CHERI that you feel should be enhanced and/or changed?
Please also share reasons for this.

Generally, more progress in porting applications and packages would be the best way to enhance the CHERI experience. It would be good to get an IDE to run natively on CheriBSD on a Morello board, especially one which supports coding in the CHERI C/C++ dialects. A potential way to do this would be to create a CHERI C/C++ extension for VSCode or another popular IDE.

22. Do you plan to continue working with CheriBSD/Morello after the end of the programme?

If so, how might you use the technologies in your own business in the future? If not, why?
We are open to ongoing work with CheriBSD or Morello after the end of the programme. Clarity on the place of CHERI within ARM's technology roadmap would help inform our decision-making. We would also evaluate Morello + Linux support when planning any future activities.

23. What feedback would you give the programme as a whole? How could this programme have been improved? Please provide some feedback on the points below:
Application process: Straightforward and lightweight process. Slight initial confusion between DSbD and Digital Catapult materials which were similar but not identical. **Onboarding:** Onboarding was smooth with thorough documentation and good support from the Digital Catapult team. **Peer to peer sessions (format, topics, etc):** The topics were very interesting and covered the information in good depth. The frequency of the sessions was about right at once a month. The length of the sessions was about right at an hour long. Perhaps it would have been better to have a consistent day/time for the sessions each month. It was very useful to have the recordings sent as I could not attend most of the sessions live. **1:1 sessions (format, support level, etc):** The Digital Catapult team were able to answer any questions we had during these meetings. The questions we were asked were sensible and the communication as a whole was to a high standard. A good effort was made to keep the meetings within the allotted time slot and not overrun. **Communications:** The Slack group was a useful resource for communicating with other cohort members. It might be a good idea to create an announcements channel which would get a post for things

such as the demo day, interim/final report with deadlines, etc.

Other:
The example CHERI exercises were good for teaching the intricacies of coding with the CHERI dialect in mind. Similarly, the documentation was fairly comprehensive for setting up the Morello board. However, it may be a good idea to make instructional videos to complement the documentation as these tend to be easier to follow, especially for less experienced programmers.

24. Would you be happy for these results to be included in research published by the University of Cambridge on the impacts of porting code to CHERI C/C++?
If so, may the name of the software target of your project be included in this research?

Yes ▾ Yes