09 April 2025

CHERI

IMPLEMENTATION OF CHERI CAPABILITIES IN A SAFETY-CRITICAL REAL-TIME OPERATING SYSTEM FOR INTELLIGENT EDGE SYSTEMS

Dmytro Yeliseyev
Software Architect, Wind River

Agenda

- Approach
- Source of inspiration
- VxWorks
- Conclusion

Approach

Approach

Due to the **complexity** of the overall system architecture and dependencies of system components, it was decided to take an incremental development approach involving **smaller steps** that would enable progress to be assessed and validated, which would reduce overall technical risk compared to attempting to integrate modifications of multiple system architecture components in a single step.

- Get VxWorks RTOS running on Morello silicon but without enabling support for CHERI capabilities.
  - Get the VxWorks RTOS kernel running in hybrid mode.
    - Enable the pure capability mode support only in VxWorks user space.

      While estimating the changes needed in the kernel running in the hybrid mode to support pure capability mode in the user space, it was found that this effort is comparable to the effort needed to run the entire kernel in pure capability mode. It was therefore decided to skip this step.

    - Enable pure capability mode support in the VxWorks kernel.

Source of inspiration

SOURCE OF INSPIRATION

UNIVERSITY OF CAMBRIDGE https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/

A script to build and run CHERI-related software—one build tool to rule them all: **cheribuild** https://github.com/CTSRD-CHERI/cheribuild

Supported operating systems include Ubuntu.

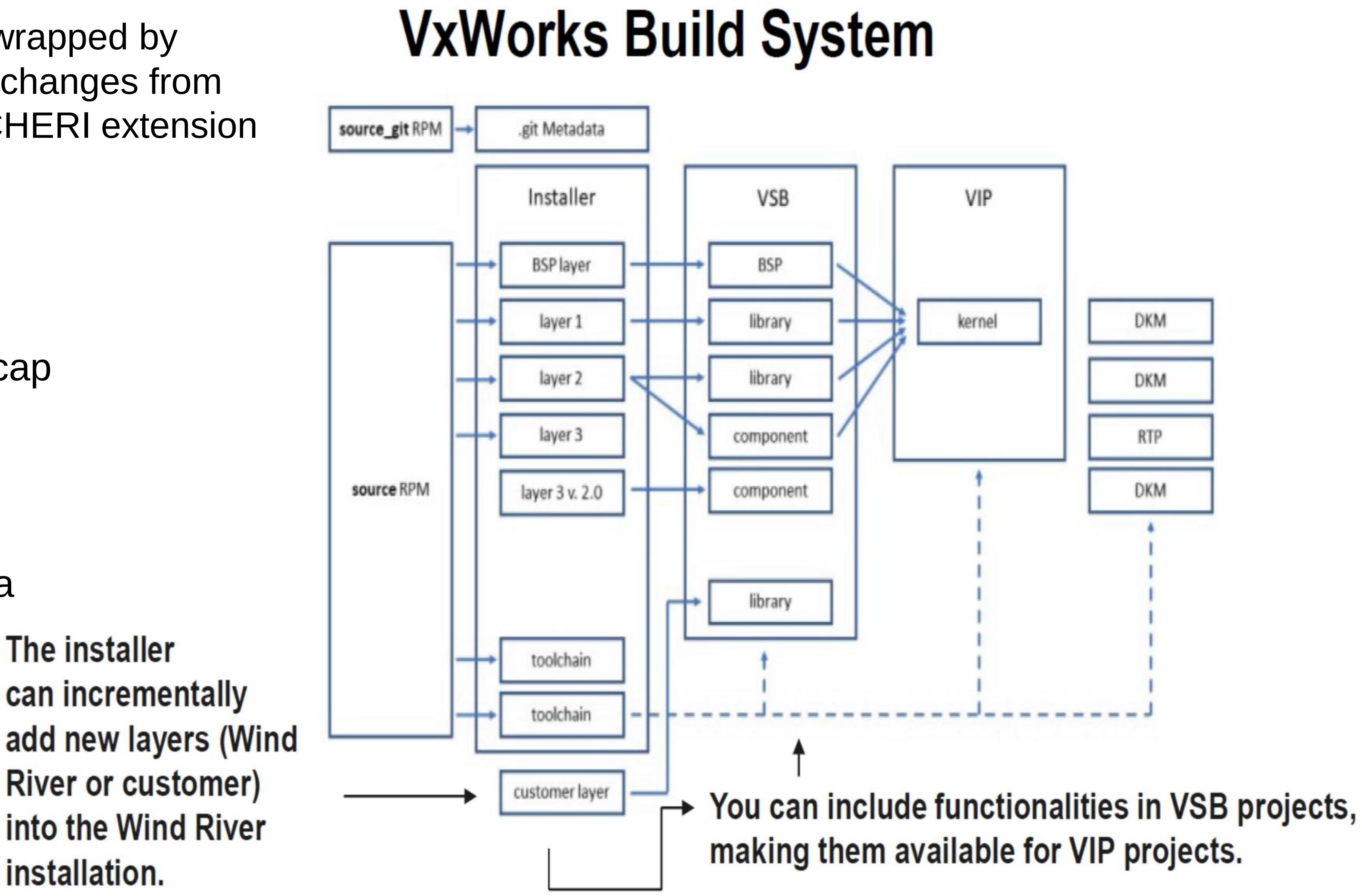
- **CheriBSD**: A complete memory- and pointer-safe FreeBSD C/C++ kernel + user space, which is very useful to get examples of how to use the CHERI software and tools existing so far.
- The **Morello SoC** is a prototype silicon implementation of a capability hardware CPU instruction set architecture (ISA): an **experimental** application of CHERI ISAv8 to ARMv8-A. The Morello SoC is based on the **Arm Neoverse N1** core with tagged memory support.
- **ARM Development Studio (Morello Edition)** can be configured to use the embedded JTAG probe on the ARM Morello SDP.

Adversarial CHERI exercises and missions: https://ctsrd-cheri.github.io/cheri-exercises

VxWorks

VxWorks – build system

- **wr-llvm-morello** - an ***LLVM*** tool chain wrapped by the ***wr-llvm*** environment and containing changes from the ***morello-llvm*** project implementing CHERI extension for the ARM8A architecture.
- **--target=arm64 -> --target=aarch64**
  - -march=morello+noa64c
  - -march=morello+a64c
  - -march=morello+c64 -mabi=purecap
- **ldarm64 -> ld.lld**
  - .cpu_private (DSECT) -> .cpu_private (COPY)
  - __cap_reloc – split .text vs .rodata
  - .size for asm symbols
  - …

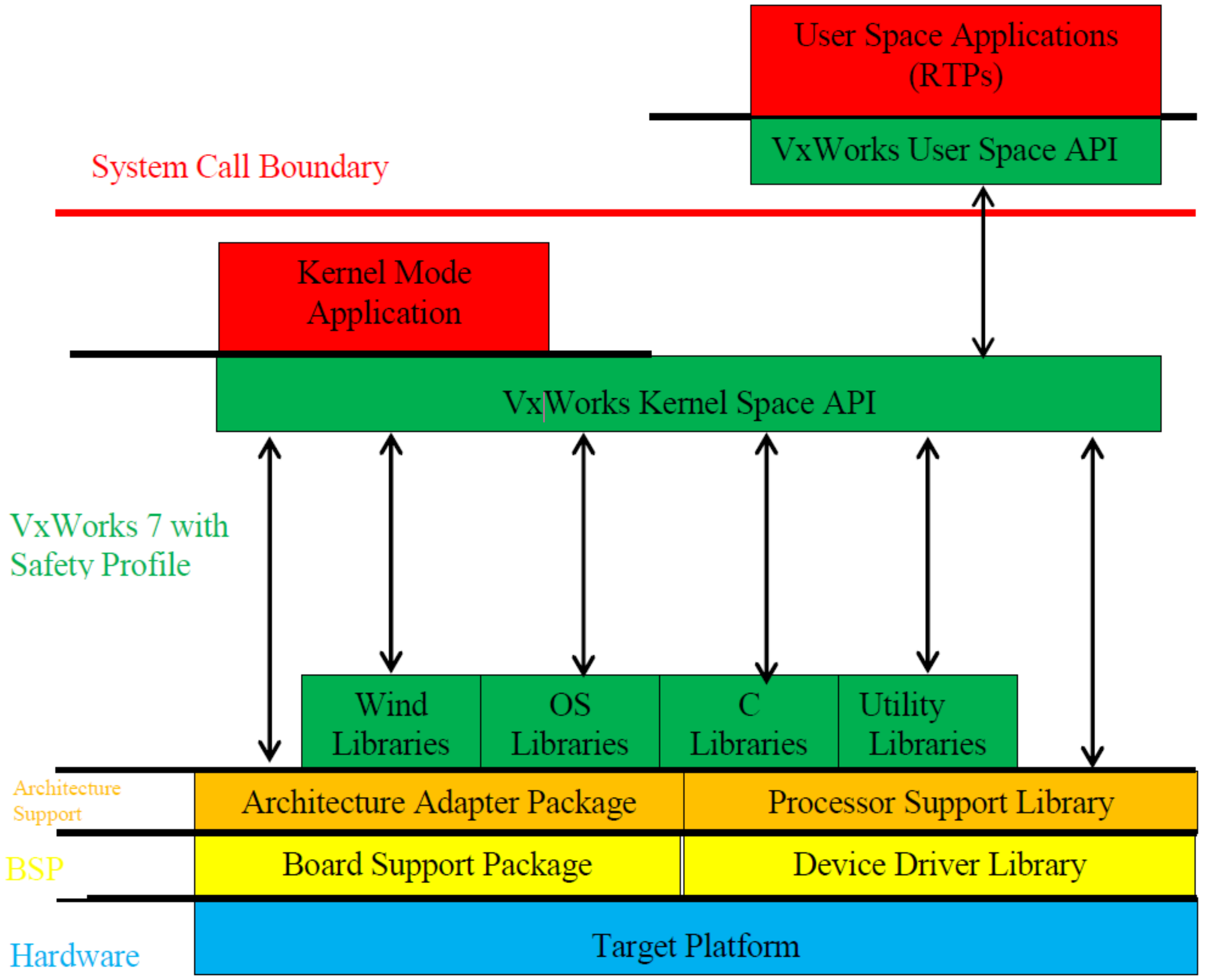

VxWorks Build System

The installer can incrementally add new layers (Wind River or customer) into the Wind River installation.

You can include functionalities in VSB projects, making them available for VIP projects.

VxWorks – RTOS components

- **HW Support - Morello SDP + QEMU :**
  - Architecture support Neoverse N1 CPU.
  - BSP + PSL (*FDT, boardLib, std drivers*)
  - MMU *(> 512GB mem addr space, etc.*)
- **Startup**
  - Vectors
  - MMU enable RW of capabilities
  - Enable CHERI instructions
  - __cap_reloc runtime initialization
- **Scheduler**
  - Extend TCBs, 128bit regs + special regs etc.
  - Align structures, system call APIs, etc.
- **Exceptions**
  - E.g. ERET required CELR instead of ELR
  - New exception types -> handlers
- **Memory Managers**
- **Kernel libraries API**
  - Tasks, Signals, Utils, Shell, User Space…
- **User Space**
  - RTP DLL: TLS descriptor reloc types support



VxWorks: SOURCE CODE

Expected problems:

Alignment issues: Capabilities are always naturally aligned. This is a requirement of the hardware.
(there is one **tag** bit per 128 bits/16 bytes)

```
/*
 * REG_SET - ARM Register set
 */

typedef struct          /* REG_SET - ARM register set */
    {
    _Vx_ULONG   r[_GREG_NUM];   /* general purpose registers */
    _Vx_ULONG   sp;             /* stack pointer             */
    _Vx_INSTR * pc;             /* program counter           */
```

```
#if __has_feature(capabilities)
               typedef uintcap_t ARM_REG_TYPE;
#else
               typedef uintptr_t ARM_REG_TYPE;
#endif

#define ARM_REG_ALIGN    _Alignas (sizeof (ARM_REG_TYPE))

#define ARM_REG_M        ARM_REG_ALIGN ARM_REG_TYPE

/*
 * REG_SET - ARM Register set
 */

typedef struct          /* REG_SET - ARM register set */
    {
    ARM_REG_M   r[_GREG_NUM];   /* general purpose registers */
    ARM_REG_M   sp;             /* stack pointer             */
    ARM_REG_M   pc;             /* program counter           */
```

bcopy: To be able to copy memory blocks with capabilities inside, you must use capability load and store instructions to propagate capability metadata and tags.

- The source address must be 16-byte aligned before whole 16-byte chunks are copied, so copy small chunks first until the address is aligned.
- Modify copy instructions:

```
ldp  x1, x2, [x0], #16        ldr  c1, [c0], #16
stp  x1, x2, [x0], #16        str  c1, [c0], #16
```

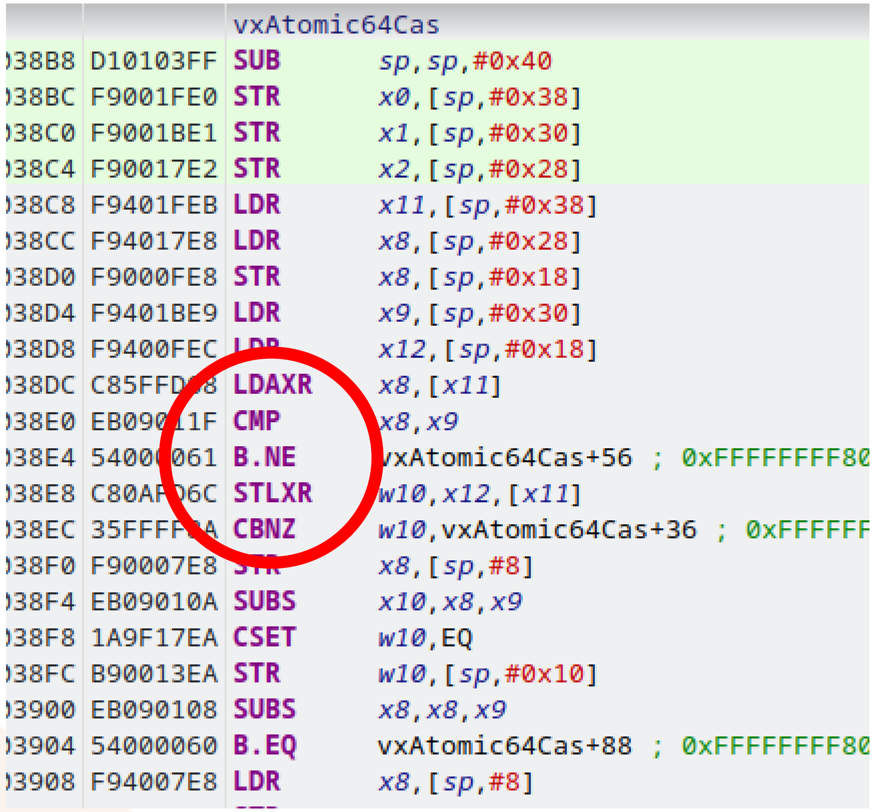
CHERI

VxWorks: SOURCE CODE

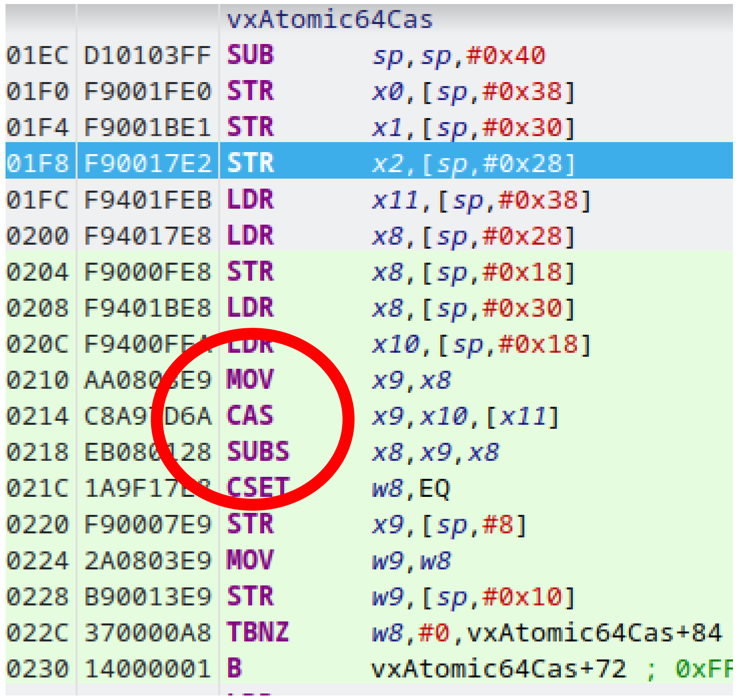
- **Unexpected problems**

Atomic op:
Non-morello: `LDAXR/STLXR`;

Morello: `CAS` – crash without ISB in front of it



```
                  vxAtomic64Cas
38B8 D10103FF SUB     sp,sp,#0x40
38BC F9001FE0 STR     x0,[sp,#0x38]
38C0 F9001BE1 STR     x1,[sp,#0x30]
38C4 F90017E2 STR     x2,[sp,#0x28]
38C8 F9401FEB LDR     x11,[sp,#0x38]
38CC F94017E8 LDR     x8,[sp,#0x28]
38D0 F9000FE8 STR     x8,[sp,#0x18]
38D4 F9401BE9 LDR     x9,[sp,#0x30]
38D8 F9400FEC LDR     x12,[sp,#0x18]
38DC C85FFD68 LDAXR   x8,[x11]
38E0 EB09011F CMP     x8,x9
38E4 54000061 B.NE    vxAtomic64Cas+56 ; 0xFFFFFFFF80
38E8 C80AFD6C STLXR   w10,x12,[x11]
38EC 35FFFF8A CBNZ    w10,vxAtomic64Cas+36 ; 0xFFFFFF
38F0 F90007E8 STR     x8,[sp,#8]
38F4 EB09010A SUBS    x10,x8,x9
38F8 1A9F17EA CSET    w10,EQ
38FC B90013EA STR     w10,[sp,#0x10]
3900 EB090108 SUBS    x8,x8,x9
3904 54000060 B.EQ    vxAtomic64Cas+88 ; 0xFFFFFFFF80
3908 F94007E8 LDR     x8,[sp,#8]
```



```
                  vxAtomic64Cas
01EC D10103FF SUB     sp,sp,#0x40
01F0 F9001FE0 STR     x0,[sp,#0x38]
01F4 F9001BE1 STR     x1,[sp,#0x30]
01F8 F90017E2 STR     x2,[sp,#0x28]
01FC F9401FEB LDR     x11,[sp,#0x38]
0200 F94017E8 LDR     x8,[sp,#0x28]
0204 F9000FE8 STR     x8,[sp,#0x18]
0208 F9401BE8 LDR     x8,[sp,#0x30]
020C F9400FEA LDR     x10,[sp,#0x18]
0210 AA0803E9 MOV     x9,x8
0214 C8A97D6A CAS     x9,x10,[x11]
0218 EB080128 SUBS    x8,x9,x8
021C 1A9F17E8 CSET    w8,EQ
0220 F90007E9 STR     x9,[sp,#8]
0224 2A0803E9 MOV     w9,w8
0228 B90013E9 STR     w9,[sp,#0x10]
022C 370000A8 TBNZ    w8,#0,vxAtomic64Cas+84
0230 14000001 B       vxAtomic64Cas+72 ; 0xFF
```

VxWorks – problems detected in compile-time

VxWorks Source Build (VSB) in pure capability mode:

- ~115 warnings not related to capabilities
- ***~2,345 warnings related to capabilities***
- Breakdown by type of warning:
    - 2,160 (~92%): Cast from provenance-free integer type to pointer type will give pointer that cannot be dereferenced
    - 110 (~5%): Alignment problems of various types; for example, structure members
    - 67 (~3%): Implicit conversion loses capability metadata
    - 8 (0.3%): Binary expression on capability types, not clear which is source of provenance

The vast majority of warnings are indicators of **traditionally-written code**, especially when assumptions are made about arbitrarily-sized integers (that is, long) being able to store pointer values.

VxWorks – problems detected in compile-time

Resolution:

- Apply VxWorks CHERI **coding rules!**
  - C/C++: Macros throughout to handle conversion between pointer and integer values
  - ASM: Macros for registers and common operations
- Modify kernel APIs using VIRT_ADDR, where a pointer (capability) is really intended/required.
- Add support for atomic operations on capabilities (128-bit values).
- Rework structure alignment as needed to be sympathetic to capabilities.
- GOT: **__cap_reloc** runtime initialization:

```
#define SYS_BOOT_LINE_LEN 256
char bootLine[SYS_BOOT_LINE_LEN];
void* addr = (void*) &bootLine;
```

✓

```
cheri_init_globals_3();
```

```
void* addr = (void*) 0x1C090000;
```

✗ →

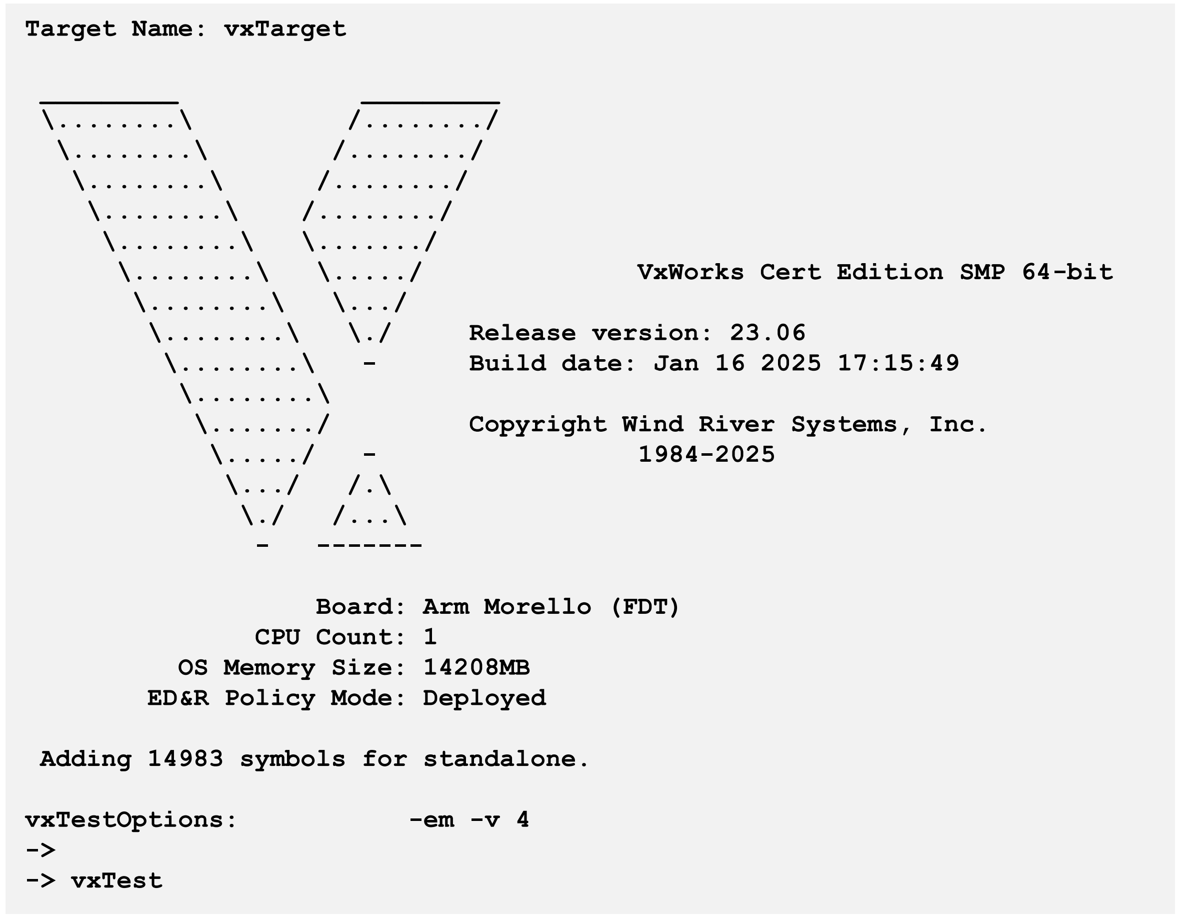
```
size_t  len = SYS_BOOT_LINE_LEN;
VIRT_ADDR const tmp = ADR_FROM_PTR (*addr);
*addr = DATA_PTR_FROM_ADR_WITH_LEN (tmp, len);
```

Conclusion

CHERI

CONCLUSION

- **Team of four engineers**—two years
- **CHERI tool chain**
- **Morello hardware and QEMU support**
- **VxTest and CHERI tests**
  - Regression test suite
  - OS integration tests
  - CHERI core functionality tests
- **Hybrid capability mode**
- **Pure capability mode**
  - Kernel: The final adjustments are in progress at the time of this material’s creation.
  - User space is coming.



```
Target Name: vxTarget


 _________            _________
 \........\          /......../
  \........\        /......../
   \........\      /......../
    \........\    /......../
     \........\   \......./
      \........\   \...../              VxWorks Cert Edition SMP 64-bit
       \........\   \.../
        \........\   \./     Release version: 23.06
         \........\   -      Build date: Jan 16 2025 17:15:49
          \........\
           \......./         Copyright Wind River Systems, Inc.
            \...../   -                 1984-2025
             \.../   /.\
              \./   /...\
               -   -------

                   Board: Arm Morello (FDT)
               CPU Count: 1
          OS Memory Size: 14208MB
        ED&R Policy Mode: Deployed

 Adding 14983 symbols for standalone.

vxTestOptions:           -em -v 4
->
-> vxTest
```