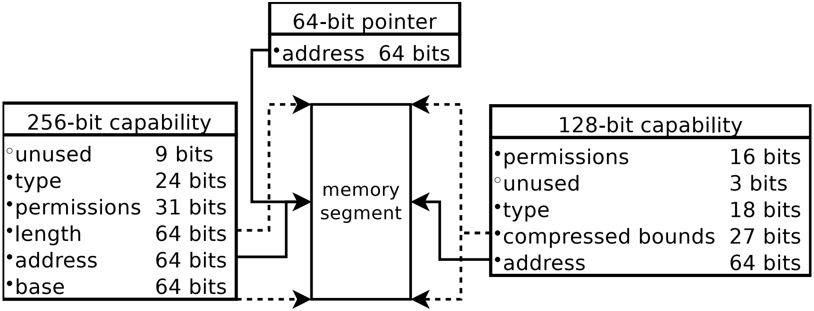
CHERI capabilities in C

Konrad Rafał Witaszczyk
def@FreeBSD.org

January 14, 2020

Pointer and CHERI capability abstractions



Supported ABIs

CheriBSD supports the following ABIs:

- Legacy ABI (freebsd32 and freebsd64);
  A process runs a program compiled for FreeBSD.
- Hybrid ABI (native ABI);
  A process partially uses capabilities.
- Pure-capability ABI (CheriABI).
  A process uses only capabilities instead of virtual addresses.

SDK

The CHERI project provides tools for FreeBSD, macOS and Linux:

1. Download, compile, install and run CheriBSD in QEMU:

```
$ git clone --single-branch --branch master \
  https://github.com/CTSRD-CHERI/cheribuild.git \
  cheri/cheribuild
$ ./cheri/cheribuild/cheribuild.py run -dv
```

SDK

2. Generate CHERI assembly code from a source file:

```
$ ./cheri/output/sdk/bin/clang \
    -S file.c \
    -target cheri-unknown-freebsd \
    -msoft-float
```

3. Cross-compile a CHERI program:

```
$ ./cheri/output/sdk/bin/clang \
    --sysroot=./cheri/output/sdk/sysroot128 \
    -B./cheri/output/sdk \
    -I./cheri/output/rootfs128/usr/include \
    -target cheri-unkown-freebsd \
    -msoft-float \
    ./file.c
```

3. Disassemble CHERI binaries:

```
$ ./cheri/output/sdk/bin/llvm-objdump -d file
```

Capability API

Kernel provides multiple symbols for capability-aware instructions that use capability registers.

```
#define cheri_getlen(x)            __builtin_cheri_length_get((x))
#define cheri_getbase(x)           __builtin_cheri_base_get((x))
#define cheri_getoffset(x)         __builtin_cheri_offset_get((x))
#define cheri_getaddress(x)        __builtin_cheri_address_get((x))
#define cheri_getperm(x)           __builtin_cheri_perms_get((x))
#define cheri_getsealed(x)         __builtin_cheri_sealed_get((x))
#define cheri_gettag(x)            __builtin_cheri_tag_get((x))
#define cheri_gettype(x)           __builtin_cheri_type_get((x))

#define cheri_andperm(x, y)        __builtin_cheri_perms_and((x), (y))
#define cheri_clearperm(x, y)      __builtin_cheri_perms_and((x), ~(y))
#define cheri_cleartag(x)          __builtin_cheri_tag_clear((x))
#define cheri_incoffset(x, y)      __builtin_cheri_offset_increment((x), (y))
#define cheri_setoffset(x, y)      __builtin_cheri_offset_set((x), (y))
#define cheri_setaddress(x, y)     __builtin_cheri_address_set((x), (y))
#define cheri_csetbounds(x, y)     __builtin_cheri_bounds_set((x), (y))

void * __capability cheri_codeptr(const void *ptr, size_t len);
void * __capability cheri_codeptrperm(const void *ptr, size_t len,
    register_t perm);
void * __capability cheri_ptr(const void *ptr, size_t len);
void * __capability cheri_ptrperm(const void *ptr, size_t len,
    register_t perm);
void * __capability cheri_ptrpermoff(const void *ptr, size_t len,
    register_t perm, off_t off);
otype_t cheri_maketype(void * __capability root_type, register_t type);
```

Buffer overflow: legacy ABI

```
int
main(int argc, char *argv[])
{
  char array[2];
  int ii;

  for (ii = 0; ii < sizeof(array) + 1; ii++) {
    array[ii] = 0;
  }

  return (0);
}
```

Buffer overflow: legacy ABI

```
$ ./cheri/output/sdk/bin/clang \
    --sysroot=./cheri/output/sdk/sysroot128 \
    -B./cheri/output/sdk \
    -I./cheri/output/rootfs128/usr/include \
    -target cheri-unknown-freebsd \
    -static \
    -msoft-float \
    -o native \
    native.c
```

Buffer overflow: legacy ABI

```
# ./native
#
```

Buffer overflow: hybrid ABI

```
#include <cheri/cheric.h>

int
main(int argc, char *argv[])
{
  char array[2];
  char * __capability arrayp;
  int ii;

  arrayp = cheri_ptr(array, sizeof(array));
  for (ii = 0; ii < sizeof(array) + 1; ii++) {
    arrayp[ii] = 0;
  }

  return (0);
}
```

Buffer overflow: hybrid ABI

```
0000000000204d8 cheri_ptr:
  204d8:   67 bd ff d0      daddiu  $sp, $sp, -48
  204dc:   ff be 00 28      sd      $fp, 40($sp)
  204e0:   03 a0 f0 25      move    $fp, $sp
  204e4:   00 a0 08 25      move    $1, $5
  204e8:   00 80 10 25      move    $2, $4
  204ec:   ff c4 00 20      sd      $4, 32($fp)
  204f0:   ff c5 00 18      sd      $5, 24($fp)
  204f4:   df c3 00 20      ld $3, 32($fp)
  204f8:   48 01 00 d3      cfromddc $c1, $3
  204fc:   df c3 00 18      ld $3, 24($fp)
  20500:   48 03 08 c8      csetbounds $c3, $c1, $3
  20504:   ff c1 00 10      sd      $1, 16($fp)
  20508:   ff c2 00 08      sd      $2, 8($fp)
  2050c:   03 c0 e8 25      move    $sp, $fp
  20510:   df be 00 28      ld      $fp, 40($sp)
  20514:   67 bd 00 30      daddiu  $sp, $sp, 48
  20518:   03 e0 00 08      jr      $ra
  2051c:   00 00 00 00      nop
```

Buffer overflow: hybrid ABI

```
$ ./cheri/output/sdk/bin/clang \
    --sysroot=./cheri/output/sdk/sysroot128 \
    -B./cheri/output/sdk \
    -I./cheri/output/rootfs128/usr/include \
    -target cheri-unknown-freebsd \
    -static \
    -msoft-float \
    -o hybrid \
    hybrid.c
```

Buffer overflow: hybrid ABI

```
# ./hybrid
Trapframe Register Dump:
$0: 0                  at: 0x2                v0: 0                  v1: 0x2
a0: 0x7fffffeaec       a1: 0x2                a2: 0x7fffffeb88       a3: 0
     x100000000
a4: 0x400e3000         a5: 0x402008c0         a6: 0x7fffffde08       a7: 0x1000
t0: 0x1                t1: 0                  t2: 0x7fffffe454       t3: 0xdb640
s0: 0x7fffffeb88       s1: 0x1                s2: 0x7fffffeb78       s3: 0
s4: 0                  s5: 0                  s6: 0                  s7: 0
t8: 0x1                t9: 0x204d8            k0: 0                  k1: 0
gp: 0xdb640            sp: 0x7fffffeab0       s8: 0x7fffffeab0       ra: 0x2045c
status: 0x408084b3 mullo: 0; mulhi: 0; badvaddr: 0xd3c3bfd
cause: 0x48; pc: 0x20498
BadInstr:  0xe8010800 csb zero,at,0(c1)
CHERI cause:  ExcCode:  0x01 RegNum:  $c01 (length violation)
$ddc: v:1 s:0 p:0007817d b:0000000000000000 l:0000008000000000 o:0 t:-1
$c01: v:1 s:0 p:0007817d b:0000007fffffeaec l:0000000000000002 o:0 t:-1
(...)
$c31: v:0 s:0 p:00000000 b:0000000000000000 l:ffffffffffffffff o:0 t:-1
$pcc: v:1 s:0 p:00068117 b:0000000000000000 l:0000008000000000 o:20498 t:-1
Aug 30 03:07:19 qemu-cheri128-def kernel: USER_CHERI_EXCEPTION: pid 722 tid
     100046 (hybrid), uid 0: CP2 fault (type 0x32)
Aug 30 03:07:19 qemu-cheri128-def kernel: Process arguments: /tmp/hybrid
Signal 34 (core dumped)
#
```

Buffer overflow: pure-capability ABI

```
int
main(int argc, char *argv[])
{
  char array[2];
  int ii;

  for (ii = 0; ii < sizeof(array) + 1; ii++) {
    array[ii] = 0;
  }

  return (0);
}
```

Buffer overflow: pure-capability ABI

```
$ ./cheri/output/sdk/bin/clang \
    --sysroot=./cheri/output/sdk/sysroot128 \
    -B./cheri/output/sdk \
    -I./cheri/output/rootfs128/usr/include \
    -target cheri-unknown-freebsd \
    -static \
    -msoft-float \
    -mabi=purecap \
    -mabi=purecap \
    -o purecap \
    purecap.c
```

Buffer overflow: pure-capability ABI

```
# ./purecap
Trapframe Register Dump:
$0: 0                  at: 0x2                v0: 0                  v1: 0
a0: 0x1                a1: 0x4                a2: 0                  a3: 0
a4: 0                  a5: 0x1                a6: 0                  a7: 0
t0: 0x20               t1: 0                  t2: 0                  t3: 0
s0: 0x1                s1: 0                  s2: 0                  s3: 0
s4: 0                  s5: 0                  s6: 0                  s7: 0
t8: 0                  t9: 0x78               k0: 0                  k1: 0
gp: 0                  sp: 0x3ff0000          s8: 0                  ra: 0
status: 0x408084b3 mullo: 0; mulhi: 0; badvaddr: 0xd4a11ec
cause: 0x48; pc: 0x120030644
BadInstr:  0xe8020800 csb zero,at,0(c2)
CHERI cause:  ExcCode:  0x01 RegNum:  $c02 (length violation)
$ddc: v:0 s:0 p:00000000 b:0000000000000000 l:ffffffffffffffff o:0 t:-1
$c01: v:1 s:0 p:0007817d b:0000007fffffe5c8 l:0000000000000004 o:0 t:-1
(...)
$c31: v:0 s:0 p:00000000 b:0000000000000000 l:ffffffffffffffff o:0 t:-1
$pcc: v:1 s:0 p:00068117 b:0000000000000000 l:0000000120800000 o:120030644 t:-1
Aug 30 03:08:03 qemu-cheri128-def kernel: USER_CHERI_EXCEPTION: pid 723 tid
     100046 (purecap), uid 0: CP2 fault (type 0x32)
Aug 30 03:08:03 qemu-cheri128-def kernel: Process arguments: ./purecap
Signal 34 (core dumped)
#
```

Permission violation: legacy ABI

```
#include <sys/param.h>

void
foo(const char *array)
{

  *__DECONST(char *, array) = 0;
}

int
main(int argc, char *argv[])
{
  char array[2];

  foo(array);

  return (0);
}
```

Permission violation: legacy ABI

```
# ./legacy
#
```

Permission violation: hybrid ABI

```
#include <cheri/cheric.h>

void
foo(const char * __capability array)
{

  *__DECONST_CAP(char * __capability, array) = 0;
}

int
main(int argc, char *argv[])
{
  char array[2];

  foo(cheri_ptrperm(array, sizeof(array),
      CHERI_PERM_LOAD));

  return (0);
}
```

Permission violation: hybrid ABI

```
# ./hybrid
Trapframe Register Dump:
$0: 0                 at: 0                 v0: 0x7fffffeac4      v1: 0x2
a0: 0x7fffffeac4      a1: 0x2               a2: 0x4               a3: 0
     x7fffffeac4
a4: 0x4004c000        a5: 0x404008c0        a6: 0x7fffffcad8      a7: 0x1000
t0: 0xfffffffffffffe0 t1: 0                 t2: 0x7fffffd134      t3: 0
     x40088630
s0: 0x7fffffeb88      s1: 0x1               s2: 0x7fffffeb78      s3: 0
s4: 0                 s5: 0                 s6: 0                 s7: 0
t8: 0x1               t9: 0x20410           k0: 0                 k1: 0
gp: 0x48010           sp: 0x7fffffea70      s8: 0x7fffffea70      ra: 0x204bc
status: 0x408084b3 mullo: 0; mulhi: 0; badvaddr: 0xa2092cc
cause: 0x48; pc: 0x20430
BadInstr:  0xe8020000 csb zero,zero,0(c2)
CHERI cause:  ExcCode:  0x13 RegNum:  $c02 (permit store violation)
$ddc: v:1 s:0 p:0007817d b:0000000000000000 l:0000008000000000 o:0 t:-1
$c01: v:1 s:0 p:00000005 b:0000007fffffeac4 l:0000000000000002 o:0 t:-1
(...)
$c31: v:0 s:0 p:00000000 b:0000000000000000 l:ffffffffffffffff o:0 t:-1
$pcc: v:1 s:0 p:00068117 b:0000000000000000 l:0000008000000000 o:20430 t:-1
Aug 30 04:55:56 qemu-cheri128-def kernel: USER_CHERI_EXCEPTION: pid 652 tid
     100055 (hybrid), uid 0: CP2 fault (type 0x32)
Aug 30 04:55:56 qemu-cheri128-def kernel: Process arguments: /tmp/hybrid

Signal 34 (core dumped)
#
```

CHERI sandbox using libcheri(3)

```
struct cheri_object co;

co.co_codecap = codecap_create(&foo, &foo_end);
co.co_datacap = foo_datacap;

(void)libcheri_invoke(co, 0,
    0, 0, 0, 0, 0, 0, 0, 0,
    NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);
```

Libraries

1. `libcheri(3)` allows to create sandboxes using CHERI compartmentalization;
2. `libc_cheri` provides `malloc()`, `calloc()` and `realloc()` that construct bounded capabilities;
3. `procstat(1)` provides information about CHERI sandbox statistics.

Example projects adapted to CHERI

1. tcpdump;
2. PostgreSQL;
3. WebKit;
4. OpenSSH.

Thank you for your attention!