Failure-Oblivious CHERI

How to bring resilience to CHERI?

Ka Wing Li

CHERITech'25

2025-11-14

Outline

CHERI

1.1 Memory Safety

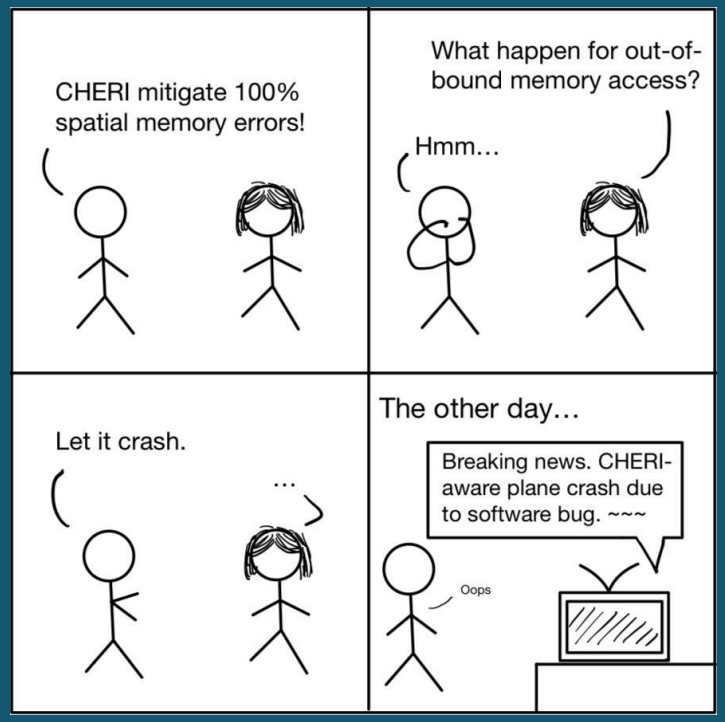
1.1.1 Spatial – e.g. Buffer Overflow

```c
uint32_t c;
uint32_t buffer[16];
uint32_t *bp = buffer;
int main() {
  assert((uintptr_t)&c == (uintptr_t)buffer + sizeof(buffer));
  memset(bp, 0x42, sizeof(buffer) + sizeof(uint32_t));
  printf("0x%08x\n", c);
  exit(EXIT_FAILURE);
}
```

1.1 Memory Safety

1.1.2 Temporal – e.g. Use After Free

```c
uint32_t *np;
int main() {
  if ((np = malloc(sizeof(*np))) == NULL) err(EXIT_FAILURE, "malloc");
  *np = 0x42; /* Allocated. */
  free(np);
  *np = 0x8; /* Freed, but in quarantine. */
  malloc_revoke_quarantine_force_flush();
  printf("0x%08x\n", *np); /* Revoked. */
  exit(EXIT_FAILURE);
}
```



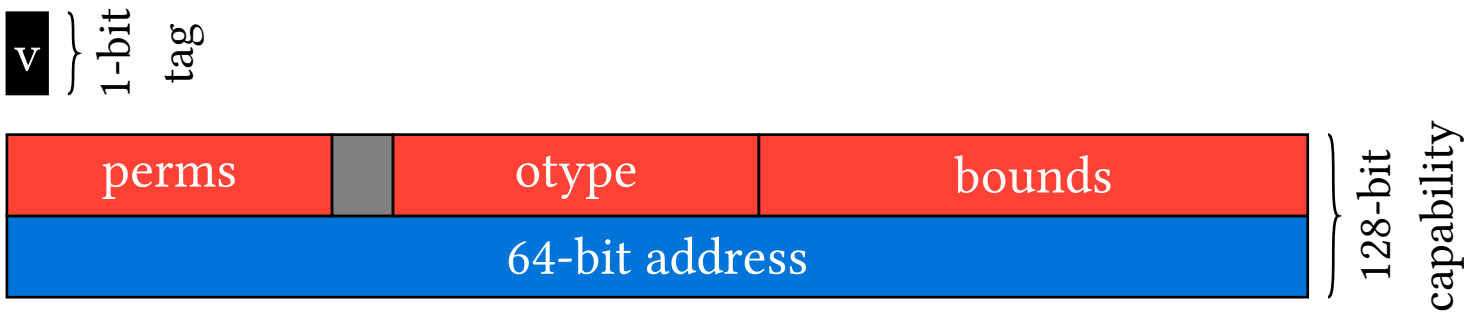
Security

vs

Reliability

1.2 Capability Hardware Enhanced RISC Instructions (CHERI)

Architectural capability in CHERI ISA version 9



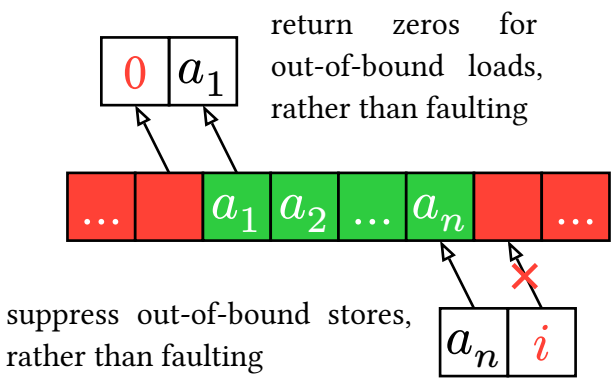
Pointer provenance valid capabilities can only be derived from existing valid capabilities

Capability monotonicity guarded derivation of new capabilities may narrow but never broaden right.

1.3 Failure-Oblivious Computing + Boundless Memory Blocks

Failure-Oblivious Computing

- in-bound access granted
- suppress out-of-bound *loads*
- suppress out-of-bound *stores*

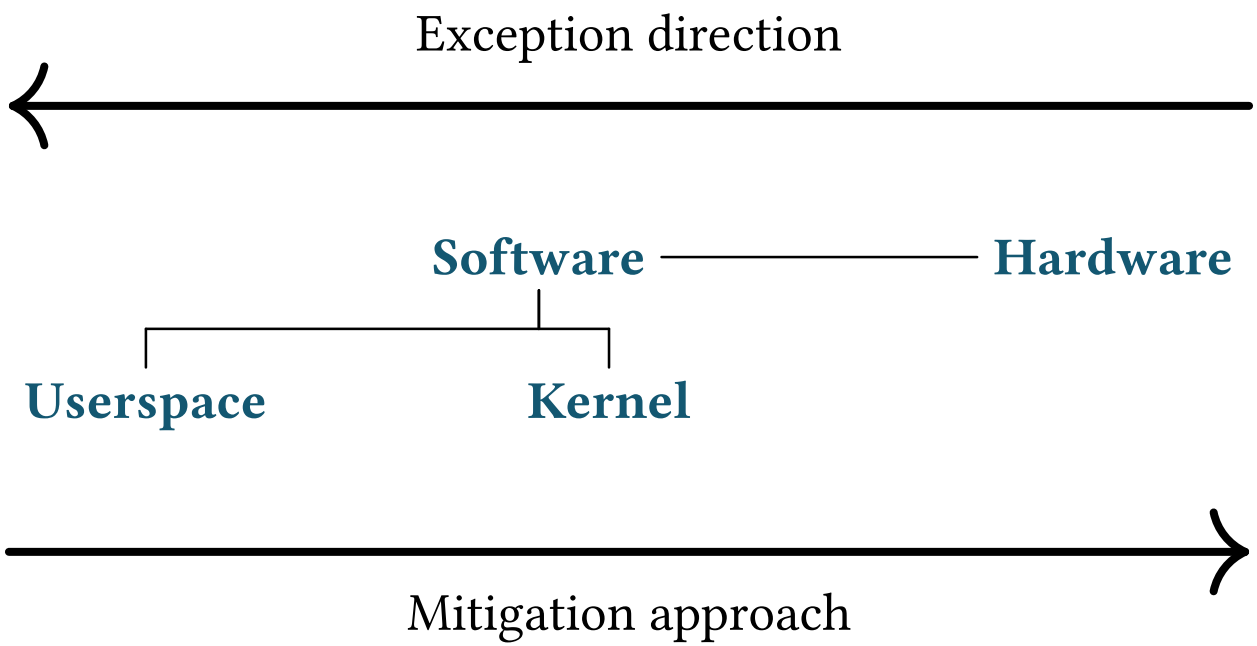


Boundless Memory Blocks

- redirect invalid memory access to LRU cache
- return manufactured (random) sequence of values for uninitialized memory

Outline

2.1 Point of Interest



2.2 From language persepective

- Null-terminated byte strings
- Off-by-one error

Bounds-checking interfaces, Annex K in the ISO C standard

```c
size_t strlen(const char *s);
size_t strnlen_s(const char *s, size_t maxsize);
char buffer[4] = "ABCD";            // missing null-terminator
printf("%d", strlen(buffer));
printf("%d", strnlen_s(buffer, 8)); // wrong size
```

Library can obtain the metadata of a capability

2.2 From language persepective

CHERI-aware libc

```c
size_t strnlen_c(const char *s, size_t maxsize) {
  auto bound = cheri_get_metadata(s);
  if (!bound.valid) return 0;
  if (bound.remain < maxsize) maxsize = bound.remain; // correct bound
  return strnlen_s(s, maxsize);
}
```

- unintended truncation?
- new C library semantics?
- runtime overhead?

2.3 From architecture persepective

- instruction-fetch failures
- load and store failures

```c
void print_secret() { printf("secret info\n"); }
char b[1024];
memcpy(b, print_secret, 1024);
void (*fp)() = (void (*)())b;
(*fp)();
```

Outline

CHERI

3.1 Buffer Overflow (again)

```c
uint32_t c;
uint32_t buffer[16];
uint32_t *bp = buffer;
int main() {
  assert((uintptr_t)&c == (uintptr_t)buffer + sizeof(buffer));
  memset(bp, 0x42, sizeof(buffer) + sizeof(uint32_t));
  printf("0x%08x\n", c);
  exit(EXIT_FAILURE);
}
```

```Shell
$ ./buffer-overflow-aarch64
# 0x42424242
$ ./buffer-overflow-cheri
# In-address space security exception (core dumped)
```

```Shell
$ LD_PRELOAD=./libcfe-ldst-preload.so ./buffer-overflow-cheri
# signo: 34, code: 1, addr: 0x1109bc
# lr: 0x402a08e9, sp: 0xffffffff7fc40, elr: 0xffffffff7fc50, ddc: 0xffffffff7fc60
# inst: 10111001000000001000000000001000
# ldst_pos
# store
# 0x00000000
```

3.2 CVE-2025-3360

A flaw was found in GLib. An *integer overflow* and *buffer under-read* occur when parsing a long invalid ISO 8601 timestamp with the `g_date_time_new_from_iso8601()` function.

```c
size_t size = 2147483747; gchar *text = (gchar *)malloc(size);
memset(text, 'a', size); text[1] = 'T'; text[size - 1] = '\0';
GDateTime *dt = g_date_time_new_from_iso8601(text, NULL);
if (dt) { gchar *formatted = g_date_time_format_iso8601(dt);
g_print("Date and Time: %s\n", formatted); g_free(formatted);
g_date_time_unref(dt); }
else { g_print("Unable to decode\n"); }
free(text);
```

```
$ ./CVE-2025-3360
# In-address space security exception (core dumped)
```

Shell

```
$ LD_PRELOAD=./libcfe-ldst-preload.so ./CVE-2025-3360
# signo: 34, code: 1, addr: 0x402dd5b0
# lr: 0x110a69, sp: 0xfffffff7fb70, elr: 0xfffffff7fb80, ddc: 0xfffffff7fb90
# inst: 00111000011110100110101100001001
# ldst_regoff
# load
# Unable to decode
```

Shell

CHERI

3.3 High-Level Programming Languages

3.3.1 Lua (CVE-2020-24371 Heap UAF)

```
1 $ _RUNTIME_REVOCATION_EVERY_FREE_ENABLE=1 lua CVE-2020-24371.lua
2 # In-address space security exception (core dumped)
3 $ LD_PRELOAD=./libcfe-ldst-preload.so
_RUNTIME_REVOCATION_EVERY_FREE_ENABLE=1 lua CVE-2020-24371.lua
4 # lua: CVE-2020-24371.lua:23: assertion failed!
5 # stack traceback:
6 #         [C]: in function 'assert'
7 #         CVE-2020-24371.lua:23: in main chunk
8 #         [C]: in ?
```

Shell

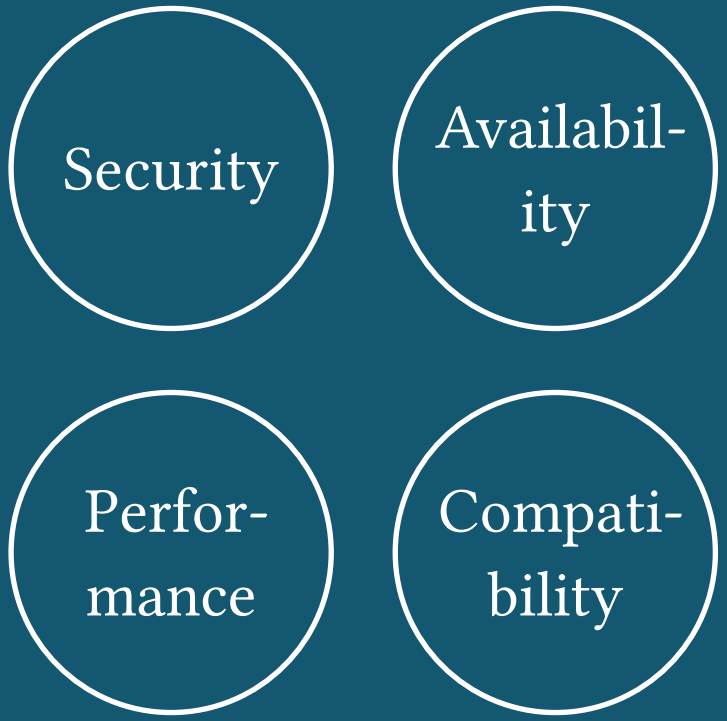
Similar result for CVE-2020-15889, CVE-2020-24369 and CVE-2020-24370.

CHERI

3.3 High-Level Programming Languages

3.3.2 Perl (CVE-2024-56406 Heap Overflow)

```shell
$ perl -e '$_ = "\x{FF}" x 1000000; tr/\xFF/\x{100}/;'
# In-address space security exception (core dumped)
$ LD_PRELOAD=./libcfe_ldst_preload.so perl -e '$_ = "\x{FF}" x 1000000; tr/\xFF/\x{100}/;'
# ...
# (exit normally)
```



- Bridge the gap between *security* and *resilience*
- Strengthen CHERI's adoption in *safety-critical* domains