Finding Container Overflow Bugs

Kostya Serebryany, Google

EuroLLVM 2014

std::vector<T> v;

- Ok: `[v.begin(), v.end())`
- Bad: `[v.end(), v.begin() + v.capacity())`
  - AddressSanitize/etc can not detect

Good news #1

```
std::vector<int> v(4);
v.reserve(8);
v[6] = 0;
```

- libc++:
  `vector[] index out of bounds`
- libstdc++ with -D_GLIBCXX_DEBUG:
  `attempt to subscript container with out-of-bounds index 6, but container only holds 4 elements.`

Good news #2

```
std::vector<int> v(4);
v.reserve(8);
auto it = v.begin();
*(it + 6) = 0; // BOOM
```

- libc++:
  `Attempted to add/subtract iterator outside of valid range`

- libstdc++ with -D_GLIBCXX_DEBUG:
  `attempt to advance a dereferenceable (start-of-sequence) iterator 6 steps, which falls outside its valid range.`

Good news #3

```
std::vector<int> v(8);
auto it = v.begin()+6;
v.resize(4);
*(it) = 0; // BOOM
```

- libc++:
  `Attempted to dereference a non-dereferenceable iterator`
- libstdc++ with -D_GLIBCXX_DEBUG:
  `attempt to dereference a singular iterator.`

Bad news

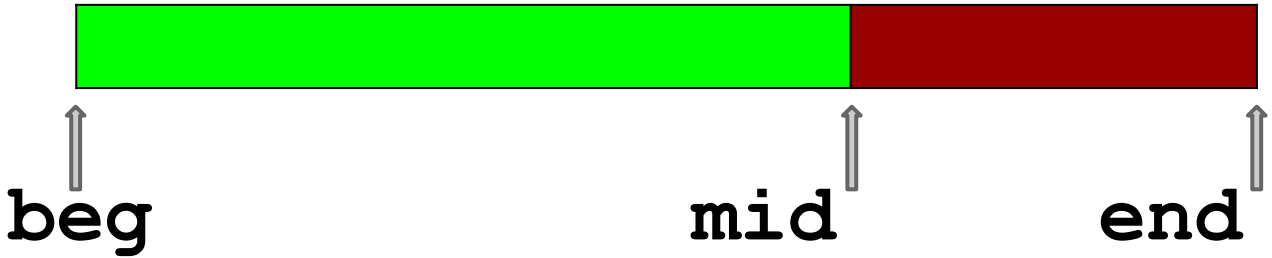
```
std::vector<int> v(4);
v.reserve(8);
int *p = v.data();
p[6] = 0;      // BOOM
```

- libc++:
  PASS
- libstdc++ with -D_GLIBCXX_DEBUG:
  PASS
- AddressSanitizer:
  PASS

Year 2014

C++ container exports its internals as a raw pointer: what a shame!

AddressSanitizer annotations



// Called on every change of size() or capacity()

```
__sanitizer_annotate_contiguous_container(
   const void *beg,      const void *end,
   const void *new_mid, const void *old_mid);
```

```
AddressSanitizer: container-overflow
WRITE of size 4 at 0x..eff8 thread T0
    #0 0x4859d7 in main t.cc:6

0x..eff8 is located 24 bytes inside
of 32-byte region [0x..efe0,0x..f000)
allocated by thread T0 here:
    #5 in main t.cc:4
```

Shadow bytes around the buggy address:

```
  0x0c067fff9da0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9db0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9dc0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9dd0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9de0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c067fff9df0: fa fa fa fa fa fa fa fa fa fa fa 00 00 fc[fc]
  0x0c067fff9e00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9e10: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9e20: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9e30: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff9e40: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
```

Improved leak detector sensitivity

```
#include <vector>
std::vector<int*> *v;
int main() {
  v = new std::vector<int*>;
  v->push_back(new int[100]);
  v->pop_back();
}
```

Current Status (std::vector<>)

- libc++:
  - Under review since Nov’13 (Ping!)
  - ~60 lines added, mostly trivial
  - Hidden under #ifdef
- libstdc++:
  - Submitted into google branch
  - Deployment in progress: 30+ bugs found

Other containers

- std::string: same story
  - need to handle small in-place strings
- std::deque: a bit different
  - problematic for types < 8 bytes since ASan requires good regions to be 8-aligned