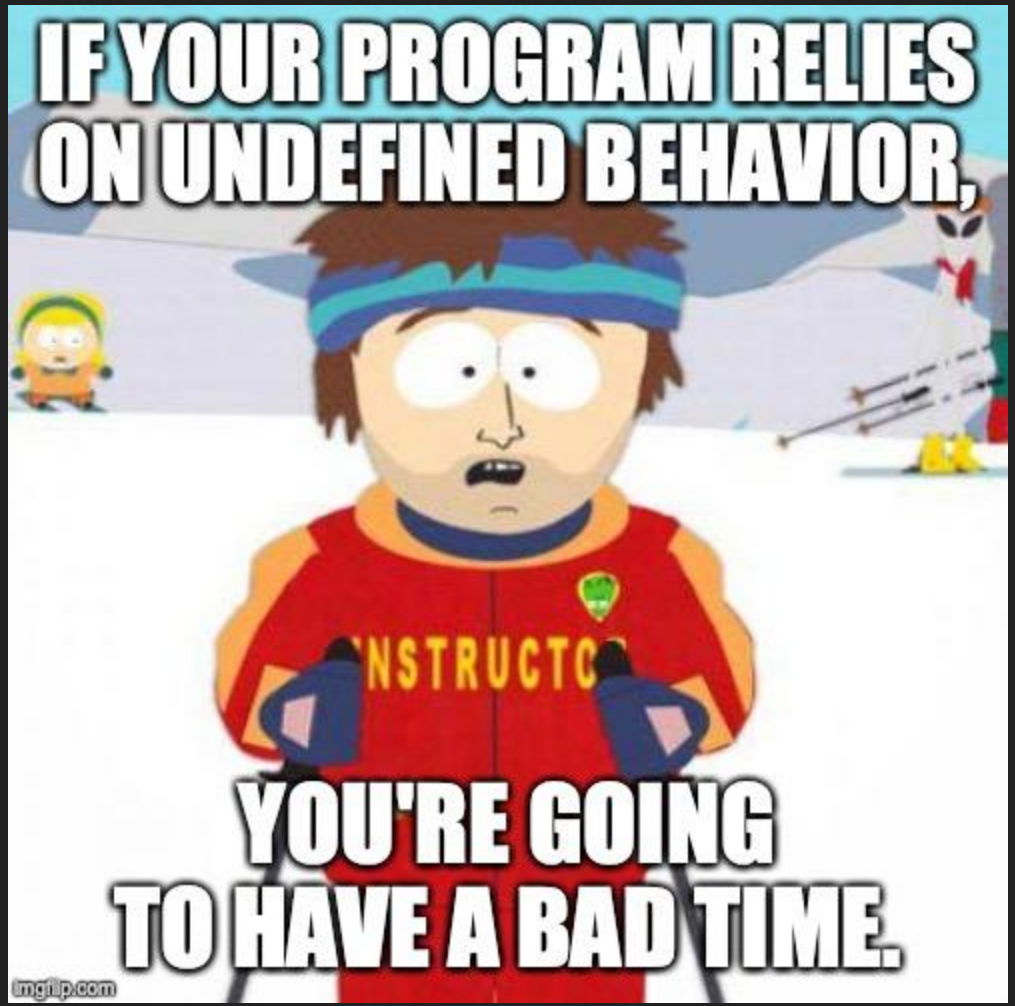
Detecting Programs That Rely on Undefined Behavior

2019/09/19

Geoffrey L. Viola


IF YOUR PROGRAM RELIES
ON UNDEFINED BEHAVIOR,
YOU'RE GOING
TO HAVE A BAD TIME.
imgflip.com

Array Out of Bounds

```
std::array<int32_t, 10> arr;

std::cout << arr[arr.size()] << std::endl;
```

Array Out of Bounds Results

```
std::array<int32_t, 10> arr;

std::cout << arr[arr.size()] << std::endl;
```

Clang-tidy

Clang Sanitizer asan

MSVC in debug mode

mktime

```
const char *const kTime = "2019";
const char *const kFormat = "%Y";
tm tmp_time;
strptime(kTime, kFormat, &tmp_time);
// proper initialization
// tmp_time.tm_isdst = -1;
time_t time = mktime(&tmp_time);
std::cout << static_cast<uint64_t>(time) << '\n';
```

mktime Results

```
const char *const kTime = "2019";
const char *const kFormat = "%Y";
tm tmp_time;
strptime(kTime, kFormat, &tmp_time);
// proper initialization
// tmp_time.tm_isdst = -1;
time_t time = mktime(&tmp_time);
std::cout << static_cast<uint64_t>(time) << '\n';
```

Clang-tidy

Valgrind

Conclusions

Run checks at multiple levels of optimization

Clang-tidy is a great

MSVC has some useful checks built in

asan+ubsan and valgrind are very useful

More information:
https://github.com/geoffviola/undefined_behavior_study