The Undefined Behavior BoF

Alexander Monakov

GNU Tools Cauldron 2018

UB Warnings

-Wub

UB Traps

-ftrapping-ub

UB Amplification

I divided by zero and now my $HOME is gone

Reckless Translation

(not-so)-fun&safe-optimizations

UB Warnings and Traps

Optimizations do not react consistently when detecting UB

- Translate as-is, *do not diagnose*
  The head-in-the-sand solution
- Translate as-is, warn
  At least we make noise?
- *Translate to trap*, warn
  Can we make a rule to prefer **this** please?

Note: not implying that optimizations should go out of their way to recognize UB in the first place. In many cases we assume that UB does not happen and do not have a cheap way to detect it.

UB Amplification

I accidentally divided by zero and now my $HOME is gone

UB that might manifest in some *benign* way $\rightarrow$ *optimization* $\rightarrow$
UB that manifests in an unexpected way

- Loop with off-by-one access $\rightarrow$ infinite loop
- memset-0 key/password buffer $\rightarrow$ no-op
- Read-only OOB access $\rightarrow$ OOB write

The "surprising" outcome may be preferable. GIGO principle applies.

Is there interest in limiting amplification effects? In a more uniform way than adding -fno options?

Together with lack of warnings they make users' lives harder.

Reckless Translation

Opinion: with stack probing GCC was on the "irresponsible" side

- **-fstack-check** was a confusing, never-used flag
- Qualys lit the proverbial fire with Stack Clash
- Would a lesser effort be able to elicit change?
- Now we have **-fstack-clash-protection** and **-fstack-check** is still confusing

Other questionable choices: libatomic, -ffast-math

Suggested BoF Agenda

1. Detecting unconditional UB in frontends and passes
   - Consistently transform to traps or poison values?
   - Avoid "head-in-the-sand" treatment?
2. Reporting to users
   - But not turning an optimizing compiler to a static analyzer
3. Should optimizations anticipate UB at all?

Questions:

- **Does this matter from QoI perspective?**
- **Can we identify desirable directions?**
- **Which issues are not important?**

Wish: Better Traps

__builtin_annotated_trap ("file:line: reason")

```
.text
        mov  %r0, $.LC0
        hcf
.rodata
.LC0:
        .string "file:line: reason"
```

`hcf` stops the program as if by POSIX abort()