Detection of undefined behavior using KLEE

Pavel Iatchenii

What is undefined behavior

- **Not** unspecified behavior
- **Not** implementation-defined behavior
- *"Behavior, upon the use of a non-portable or erroneous program construct or of erroneous data, for which International Standard imposes no requirements"* (C99 standard)
- *"Anything at all can happen; the Standard imposes no requirements. The program may fail to compile, or it may execute incorrectly..."* (comp.lang.c)

What is unintentional behavior

This is well-defined behavior, opposite to undefined behavior, which usually goes against programmers' intent and may also be a bug.

UB in symbolic execution

- Injection of checks by KLEE: division by zero, overshift overflow
- Natural processing by KLEE: dereferencing a nullptr, reaching an unreachable program point, etc
- Cases that are hard to catch without code instrumentation: integer overflow, use of a misaligned pointer, etc

How it works in LLVM right now

LLVM UndefinedBehaviorSanitizer consists of several parts:

- Code generator, uses compile-time instrumentation to insert certain kinds of checks along with **handlers**
- Runtime, implements **handlers** and exits the program if so configured

How much work has been done in KLEE

KLEE version of UB detector consists of several parts:

- **Unchanged** LLVM code generator to insert **handlers**
- **Adopted** LLVM runtime to accurately analyse the passed arguments containing source location and values of **handlers**
- **Custom** tests with symbolic variables for different types of UB

How to start detecting UB

- Build bitcode with ***-fsanitize=**** sanitizer options of your choice
- Run KLEE, the rest is done by KLEE runtime itself
- **NEW!** It is now possible to detect cases of UB in the next poster examples and many others, check out LLVM docs to explore more

References

https://github.com/klee/klee/pull/1378
https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html
https://utbot.org

Examples of undefined behavior

Signed integer oveflow

```
signed int sum(signed int x, signed int y) {
  return x + y;
}
```

If sum of `x` *and* `y` *exceeds* 2147483647*.*

Pointer oveflow

```
char access(char *ptr, int offset) {
  return *(ptr + offset)
}
```

If `ptr` *is a nullptr or the calculation blows past the end of address space.*

Usage of invalid builtin

```
int ctz(unsigned int x) {
  return __builtin_ctz(x);
}
```

If `x` *is zero.*

Use of misaligned pointer

```
char *pass(__attribute__((align_value(4))) char *ptr)
  return ptr;
}
```

If `ptr` *is not aligned to* 4 *bytes.*

Examples of unintentional behavior

Unsigned integer oveflow

```
unsigned int sum(unsigned int x, unsigned int y) {
  return x + y;
}
```

If sum of `x` *and* `y` *exceeds* 4294967295*.*

Implicit truncation

```
unsigned char convert(signed int x) {
  return x;
}
```

If that results in data loss.

Violation of nullable attribute

```
char *_Nonnull pass(char *ptr) {
  return ptr;
}
```

If `ptr` *is a nullptr.*