Dynamic Analysis in Practice

Miroslav Franc ▪ miroslav.franc@nic.cz ▪ October 7, 2018

CZ DOMAIN REGISTRY

Who am I?

- Developer of a C++ backend (FRED)
  - Free Registry for ENUM and Domains
  - https://fred.nic.cz/

What is dynamic analysis?

	source	binary
static	splint	...
dynamic	...	valgrind

How it works roughly?

- valgrind is an emulator
- sanitizers are parts of a compiler infrastructure

A bit of history?

- ElectricFence
- mudflap
- Purify
- PIN

Why not static analysis?

- false positives
- good to have real bugs with real reproducers

Types of Errors

Memory errors

- leaks
- out-of-bound access
- use-after-free
- use-after-return
- unitialized memory

Leaks?

```
{
    ...
    goto error; // jump to a cleanup label
    ...
    error:
        free(foo);
}

{
    Foo foo; // RAII
    ...
}
```

Really?

```c
#include <stdlib.h>
#include <stdio.h>

void release(void *p) { free(*(void **)p); }
#define scoped_ptr(type, n, name) \
 __attribute__((cleanup(release))) type * name = \
 calloc(sizeof(type), n)

int main(void)
{
    const size_t size = 10;

    scoped_ptr(int, size, lots_of_ints);
}
```

But we can do funny stuff in C++ as well

- std::shared_ptr cycles
- std::unique_ptr's get() and release()
- accidental temporaries

Multi-threaded errors

- data races
  - two threads access a shared memory and at least one is write
- deadlocks
- lock contention

Other kinds of Undefined behavior

- signed overflow
- null pointer dereference
- misaligned pointer dereference
- divide by zero
- load of out-of-range bool or enum value
- VLA size is negative

VLA?

```
int foo(int n)
{
    char array[n];
    ...
}

int foo()
{
    folly::small_vector<char, 1024> vec;
    ...
}
```

memcpy vs. memmove

- restrict keyword
- memcheck or memstomp

Valgrind

- english speakers tend to get the name wrong
- not just memcheck, it is more like a framework
  - memcheck, drd, helgrind...

Address Sanitizer

- (-fsanitize=address)
- out-of-bound access (heap, stack, global ojects)
- use-after-free
- less than 2x times slower
- red-zones
  - compiler - stack, global objects
  - run-time library - heap

ODR violation

- ASAN_OPTIONS=detect_odr_violation=1

Leak Sanitizer

- (-fsanitize=leak)
- is part of address sanitizer

Memory Sanitizer

- (-fsanitize=memory)
- unitialized memory
- less than 3x times slower
- shadow memory, 1-1
- you need to recompile the world :(
- variable names

Thread Sanitizer

- (-fsanitize=thread)
- ~10x times slower
- first version was based on valgrind
- support for atomics

Bad things about Valgrind

- it is slow
- serializes threads
- memcheck does not detect stack and global objects overruns
- cannot detect some kinds of undefined behaviors because it does not know as much as the compiler
- program that is executed is different from the one fed to the valgrind

Good things about Valgrind

- it supports a huge number of architectures
- works out of the box - no recompilation needed
- handles the entire userland, including the third-party libraries

Tagging of Memory

- Hardware Assisted Address Sanitizer
- so far only clang on Aarch64
- tag pointer and associated memory block

Questions?