ThreadSanitizer APIs for External Libraries

Kuba Mracek, Apple

ThreadSanitizer

ThreadSanitizer

- Data race detector

ThreadSanitizer

- Data race detector
- LLVM IR instrumentation:

ThreadSanitizer

- Data race detector
- LLVM IR instrumentation:
  - memory reads and writes

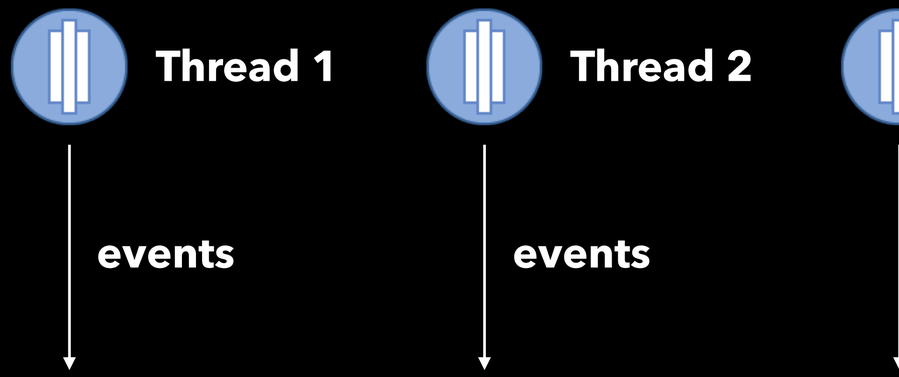
ThreadSanitizer

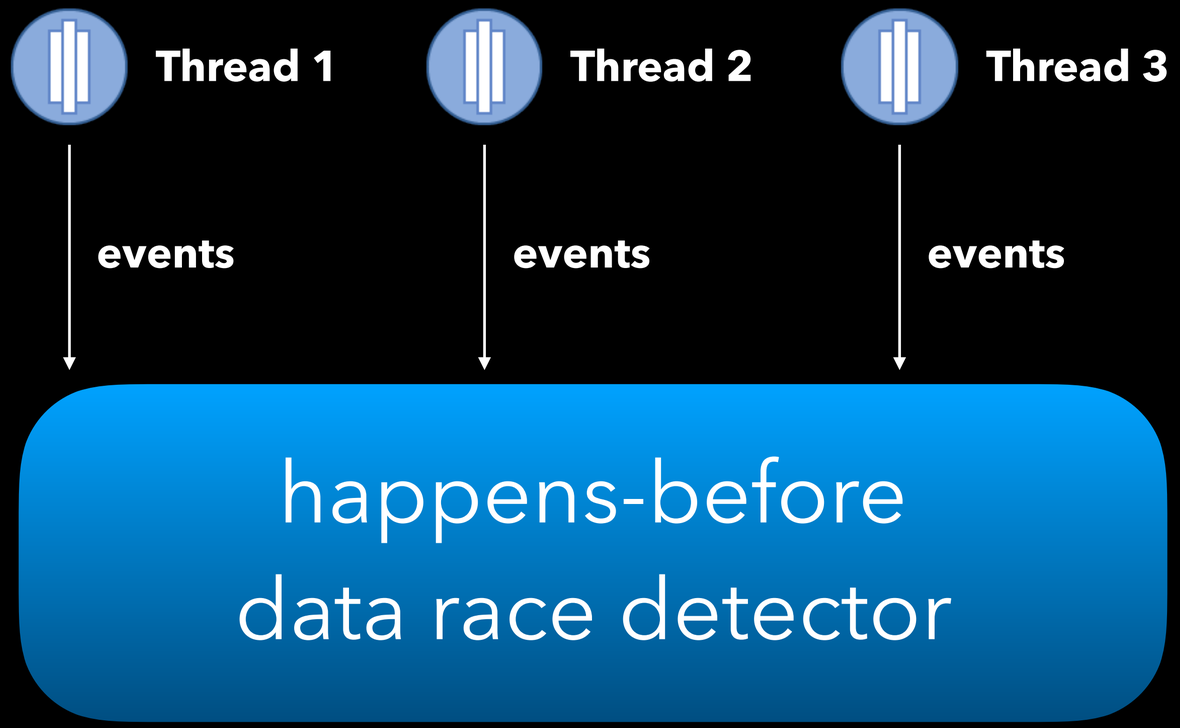
- Data race detector
- LLVM IR instrumentation:
  - memory reads and writes
  - atomic operations (load, store, RMW, CAS)

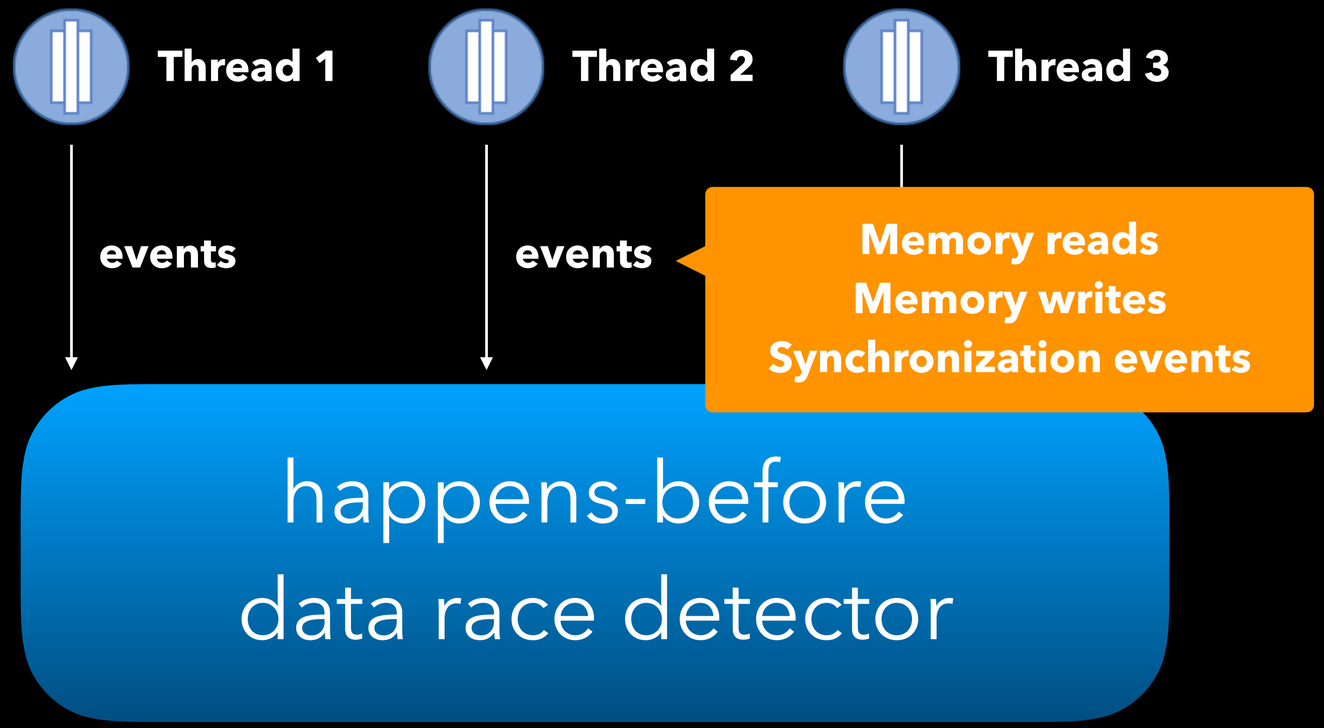

Thread 1

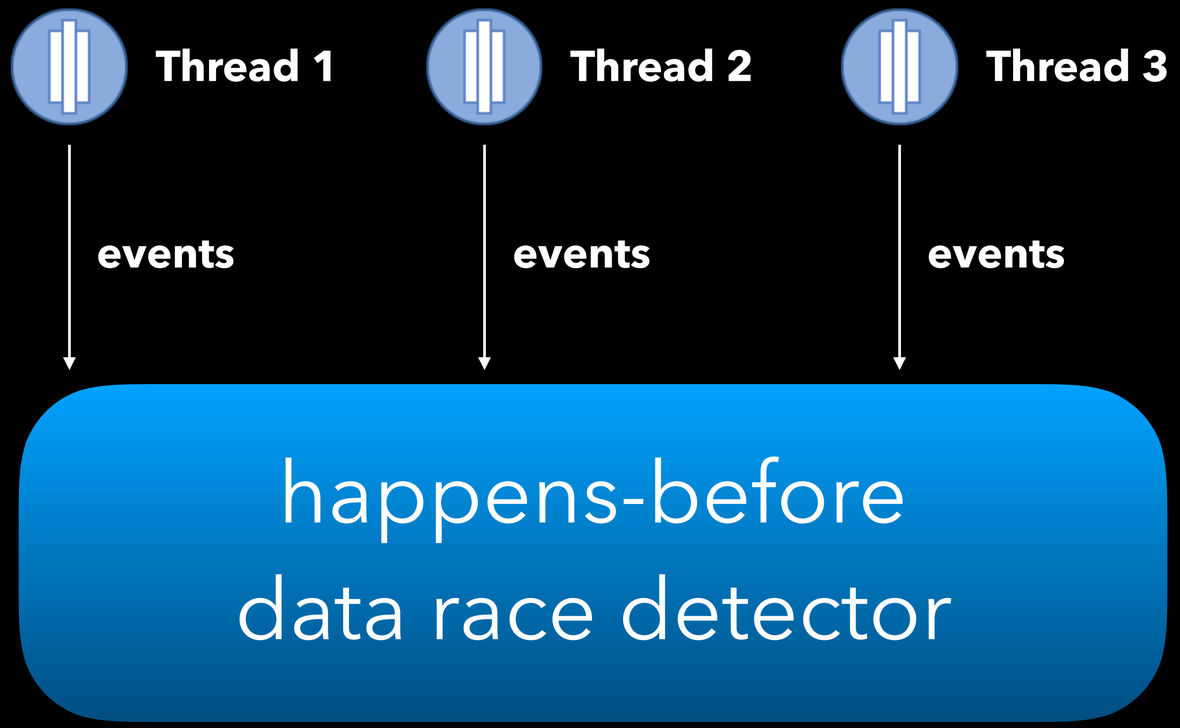
Thread 2

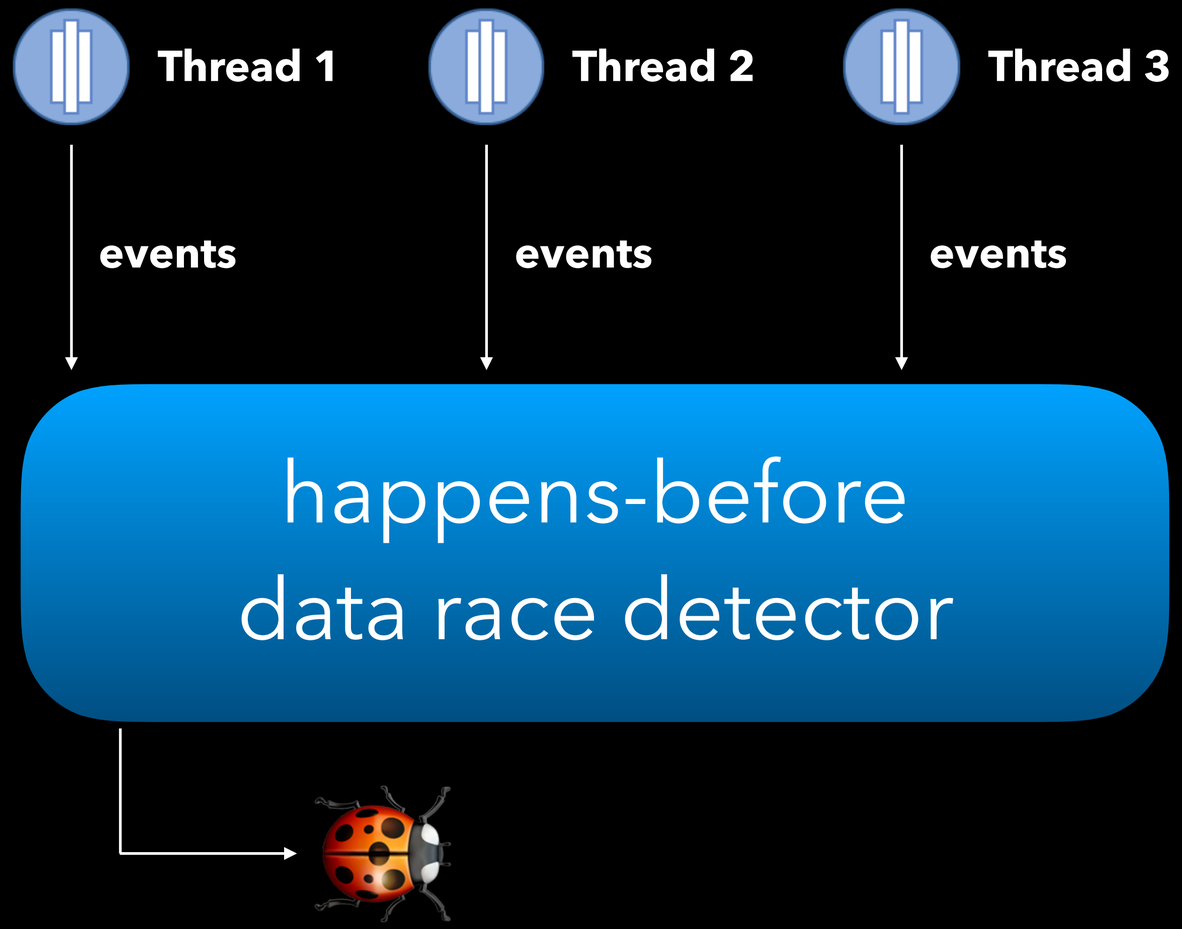
Thread 3

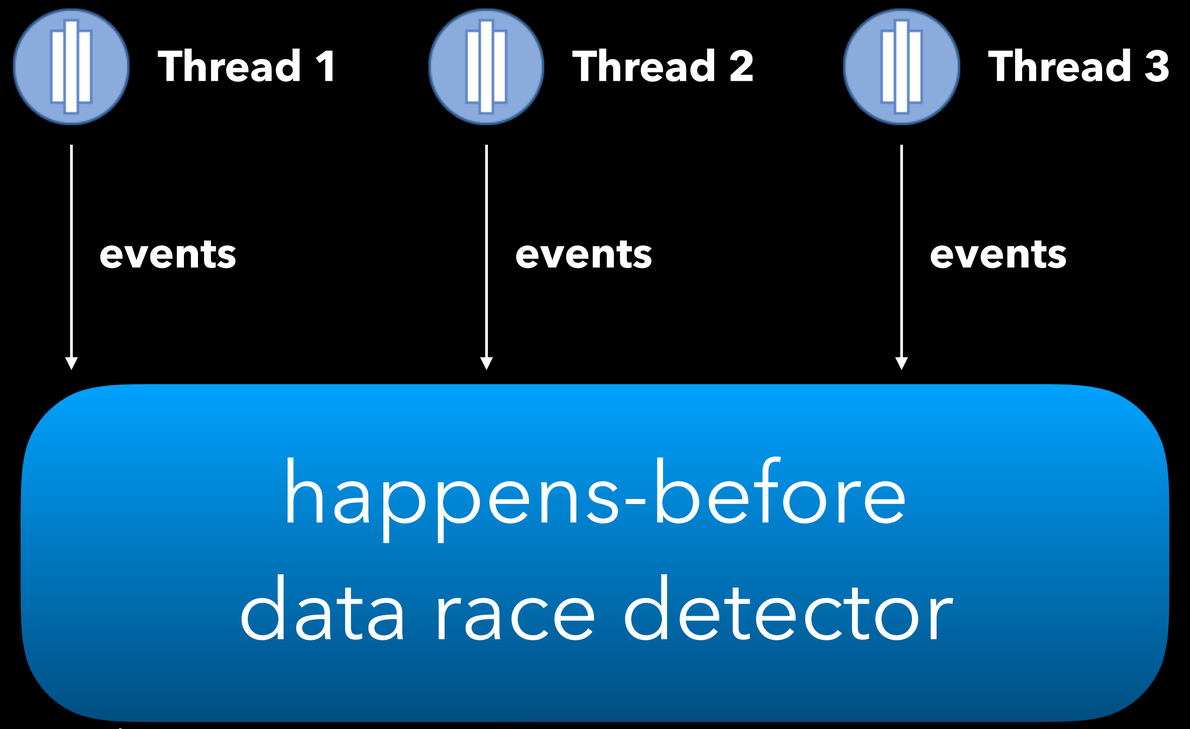

Thread 1
Thread 2
Thread 3
events
events
events


Thread 1
Thread 2
Thread 3
events
events
events
happens-before
data race detector


Thread 1
Thread 2
Thread 3
events
events
Memory reads
Memory writes
Synchronization events
happens-before
data race detector


Thread 1
Thread 2
Thread 3
events
events
events
happens-before
data race detector


Thread 1
Thread 2
Thread 3
events
events
events
happens-before
data race detector



```
$ ./racyapp
WARNING: ThreadSanitizer: data race (
  Write of size 4 at 0x7fcf47b21bc0 b
    #0 Thread1 race.c:4 (exe+0x000000

  Previous write of size 4 at 0x7fcf4
    #0 main race.c:10 (exe+0x00000000

  …
```

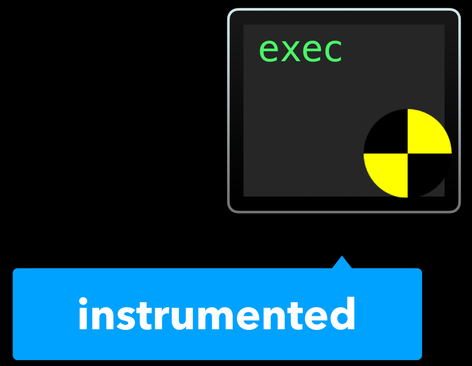
Libraries and Frameworks

Libraries and Frameworks

- Precompiled code is not instrumented

Libraries and Frameworks

- Precompiled code is not instrumented

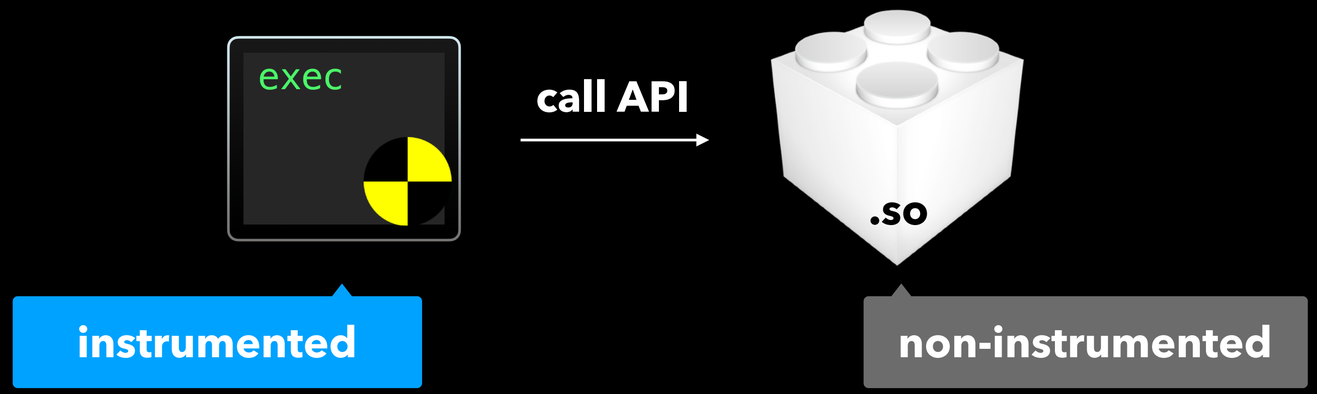


Libraries and Frameworks

- Precompiled code is not instrumented

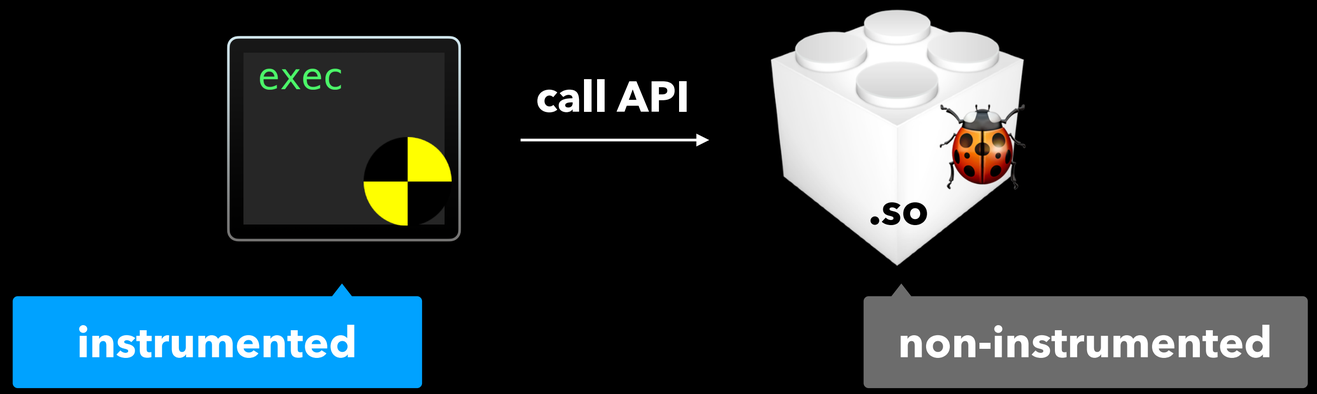
Libraries and Frameworks

- Precompiled code is not instrumented



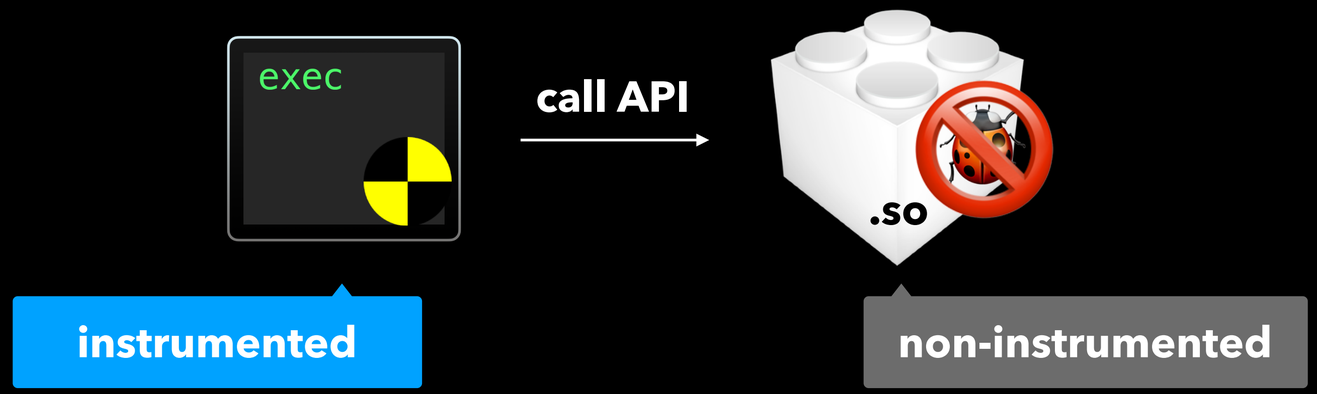
Libraries and Frameworks

- Precompiled code is not instrumented



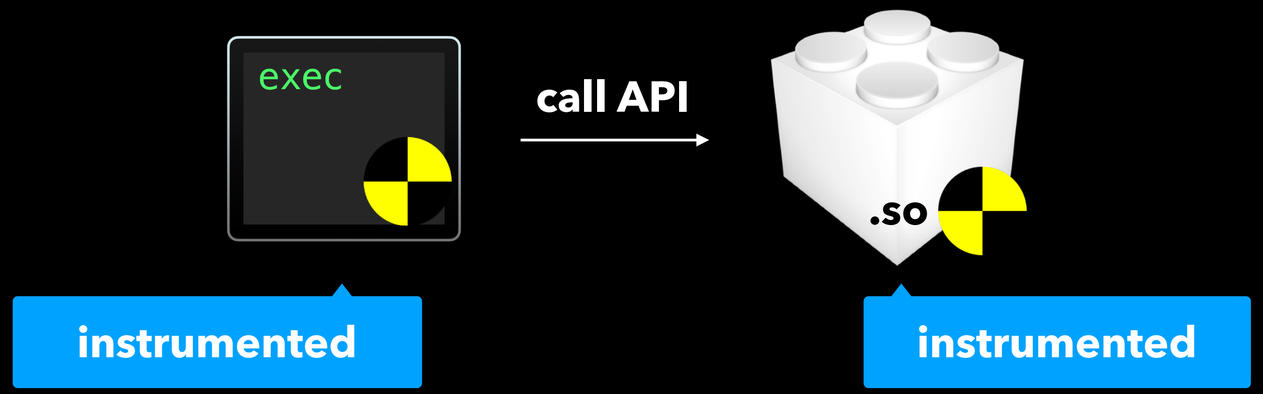
Libraries and Frameworks

- Precompiled code is not instrumented



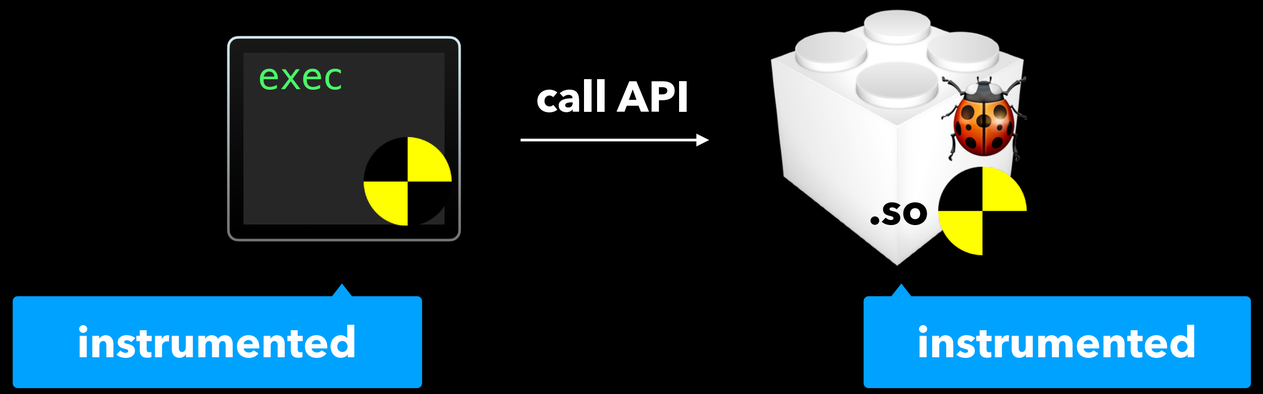
Libraries and Frameworks

- Precompiled code is not instrumented



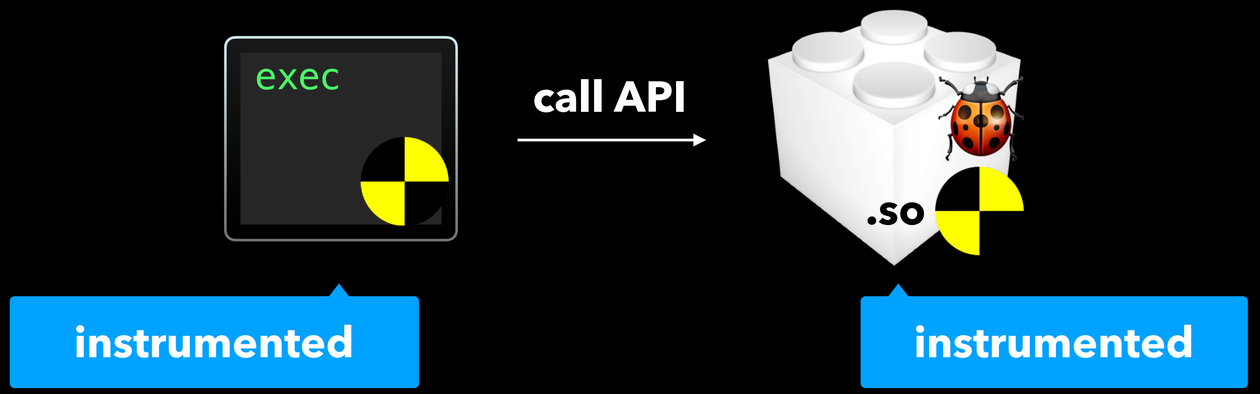
Libraries and Frameworks

- Precompiled code is not instrumented



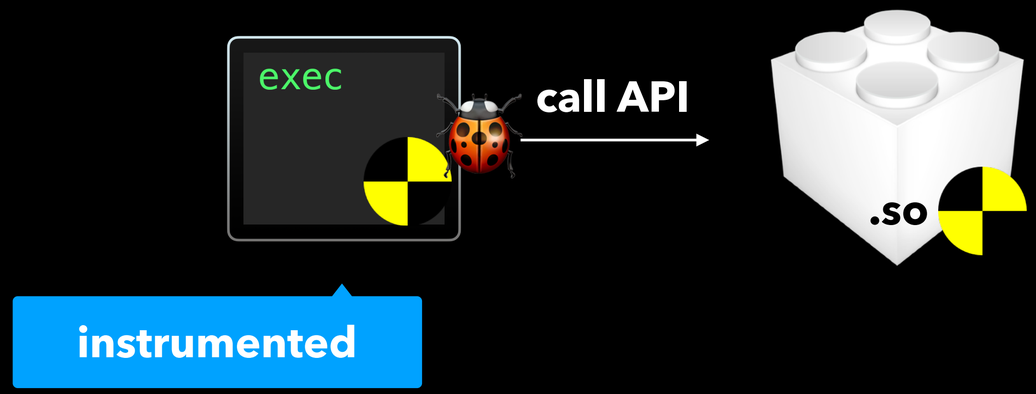
Libraries and Frameworks

- Precompiled code is not instrumented
- APIs expect users to ensure thread safety



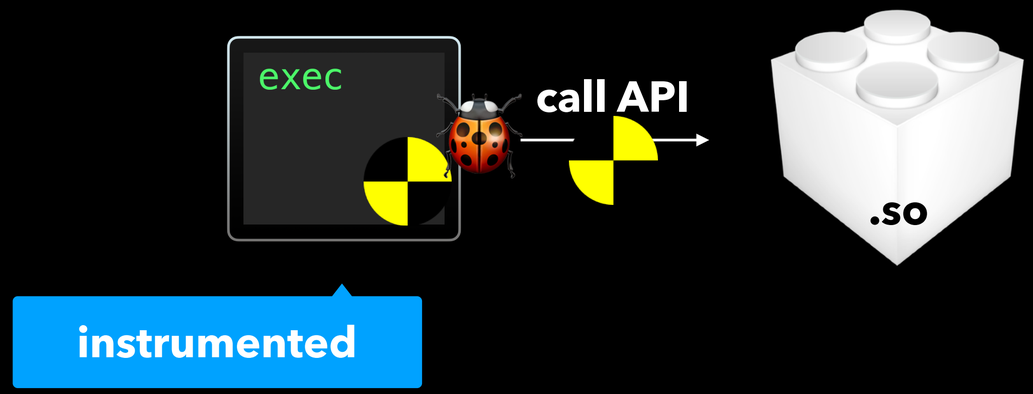
Libraries and Frameworks

- Precompiled code is not instrumented
- APIs expect users to ensure thread safety



Libraries and Frameworks

- Precompiled code is not instrumented
- APIs expect users to ensure thread safety



New: APIs for Libraries

New: APIs for Libraries

- ThreadSanitizer provides callbacks for libraries to inform about read/write-like events of high-level objects:

New: APIs for Libraries

- ThreadSanitizer provides callbacks for libraries to inform about read/write-like events of high-level objects:

```
__tsan_external_read(void *addr, void *caller_pc, void *tag);
__tsan_external_write(void *addr, void *caller_pc, void *tag);
```

New: APIs for Libraries

- ThreadSanitizer provides callbacks for libraries to inform about read/write-like events of high-level objects:

```
__tsan_external_read(void *addr, void *caller_pc, void *tag);
__tsan_external_write(void *addr, void *caller_pc, void *tag);
```

- High-level object = basically any object you work with
  - array, map, graph node, data object, UI element, …

Example: CoreFoundation

Example: CoreFoundation

- Provides APIs for basic collections:

```
CFMutableArrayRef CFArrayCreateMutable(/*…*/);
void CFArrayAppendValue(CFArrayRef array, /*…*/);
CFIndex CFArrayGetCount(CFArrayRef array);
```

Example: CoreFoundation

- Provides APIs for basic collections:

```
CFMutableArrayRef CFArrayCreateMutable(/*…*/);
void CFArrayAppendValue(CFArrayRef array, /*…*/);
CFIndex CFArrayGetCount(CFArrayRef array);
```

- User must ensure thread safety

```
// Modifies the array
void CFArrayAppendValue(CFArrayRef array, /*…*/) {
    __tsan_external_write(array, CALLER_PC, tag);
    /*…*/
}

// Reads the array
CFIndex CFArrayGetCount(CFArrayRef array) {
    __tsan_external_read(array, CALLER_PC, tag);
    /*…*/
}
```

```
// Modifies the array
void CFArrayAppendValue(CFArrayRef array, /*…*/) {
    if (is_tsan_present)
        __tsan_external_write(array, CALLER_PC, tag);
    /*…*/
}

// Reads the array
CFIndex CFArrayGetCount(CFArrayRef array) {
    if (is_tsan_present)
        __tsan_external_read(array, CALLER_PC, tag);
    /*…*/
}
```

```
==================
WARNING: ThreadSanitizer: race on a library object
  Read-only access of CFMutableArray at 0x7b0c00046b30 by
  thread T2:
    #0 CFArrayGetCount (CoreFoundation:x86_64)
    #1 Thread1 main.m:16 (demoapp:x86_64)

  Previous modifying access of CFMutableArray at
  0x7b0c00046b30 by thread T3:
    #0 CFArrayAppendValue (CoreFoundation:x86_64)
    #1 Thread2 main.m:21 (demoapp:x86_64)

  Location is heap block of size 40 at 0x7b0c00046b30
  allocated by main thread:
    …

SUMMARY: ThreadSanitizer: race on a library object
main.m:16 in Thread1
==================
```

report description

```
==================
WARNING: ThreadSanitizer: race on a library object
  Read-only access of CFMutableArray at 0x7b0c00046b30 by
  thread T2:
    #0 CFArrayGetCount (CoreFoundation:x86_64)
    #1 Thread1 main.m:16 (demoapp:x86_64)

  Previous modifying access of CFMutableArray at
  0x7b0c00046b30 by thread T3:
    #0 CFArrayAppendValue (CoreFoundation:x86_64)
    #1 Thread2 main.m:21 (demoapp:x86_64)

  Location is heap block of size 40 at 0x7b0c00046b30
  allocated by main thread:
    …

SUMMARY: ThreadSanitizer: race on a library object
main.m:16 in Thread1
==================
```

report description

type of the object

```
==================
WARNING: ThreadSanitizer: race on a library object
  Read-only access of CFMutableArray at 0x7b0c00046b30 by
  thread T2:
    #0 CFArrayGetCount (Co                    4)
    #1 Thread1 main.m:16 (demoapp:x86_64)

  Previous modifying access of CFMutableArray at
  0x7b0c00046b30 by thread T3:
    #0 CFArrayAppendValue (CoreFoundation:x86_64)
    #1 Thread2 main.m:21 (demoapp:x86_64)

  Location is heap block of size 40 at 0x7b0c00046b30
  allocated by main thread:
    …

SUMMARY: ThreadSanitizer: race on a library object
main.m:16 in Thread1
==================
```

```
==================
WARNING: ThreadSanitizer: race on a library object
  Read-only access of CFMutableArray at 0x7b0c00046b30 by
  thread T2:
    #0 CFArrayGetCount (Co                          4)
    #1 Thread1 main.m:16 (demoapp:x86_64)

  Previous modifying access of CFMutableArray at
  0x7b0c00046b30 by thread T3:
    #0 CFArrayAppendValue (CoreFoundation:x86_64)
    #1 Thread2 ma                                  86_64)

  Location is heap block of size 40 at 0x7b0c00046b30
  allocated by main thread:
    …

SUMMARY: ThreadSanitizer: race on a library object
main.m:16 in Thread1
==================
```

report description

type of the object

API call

More Details

More Details

- Tags to identify the type of the object

More Details

- Tags to identify the type of the object
- Provide caller PC

More Details

- Tags to identify the type of the object
- Provide caller PC
- Weak imports

More Details

- Tags to identify the type of the object
- Provide caller PC
- Weak imports
- Detect ThreadSanitizer at initialization time

More Details

- Tags to identify the type of the object
- Provide caller PC
- Weak imports
- Detect ThreadSanitizer at initialization time
- Contact me or thread-sanitizer@googlegroups.com mailing list

More Details

- Tags to identify the type of the object
- Provide caller PC
- Weak imports
- Detect ThreadSanitizer at initialization time
- Contact me or thread-sanitizer@googlegroups.com mailing list
- Already used by Foundation, CoreFoundation and Swift

If you’re developing a popular library used in multithreaded programs, consider adopting these APIs!