Developer Tools

#WWDC17

Finding Bugs Using Xcode Runtime Tools

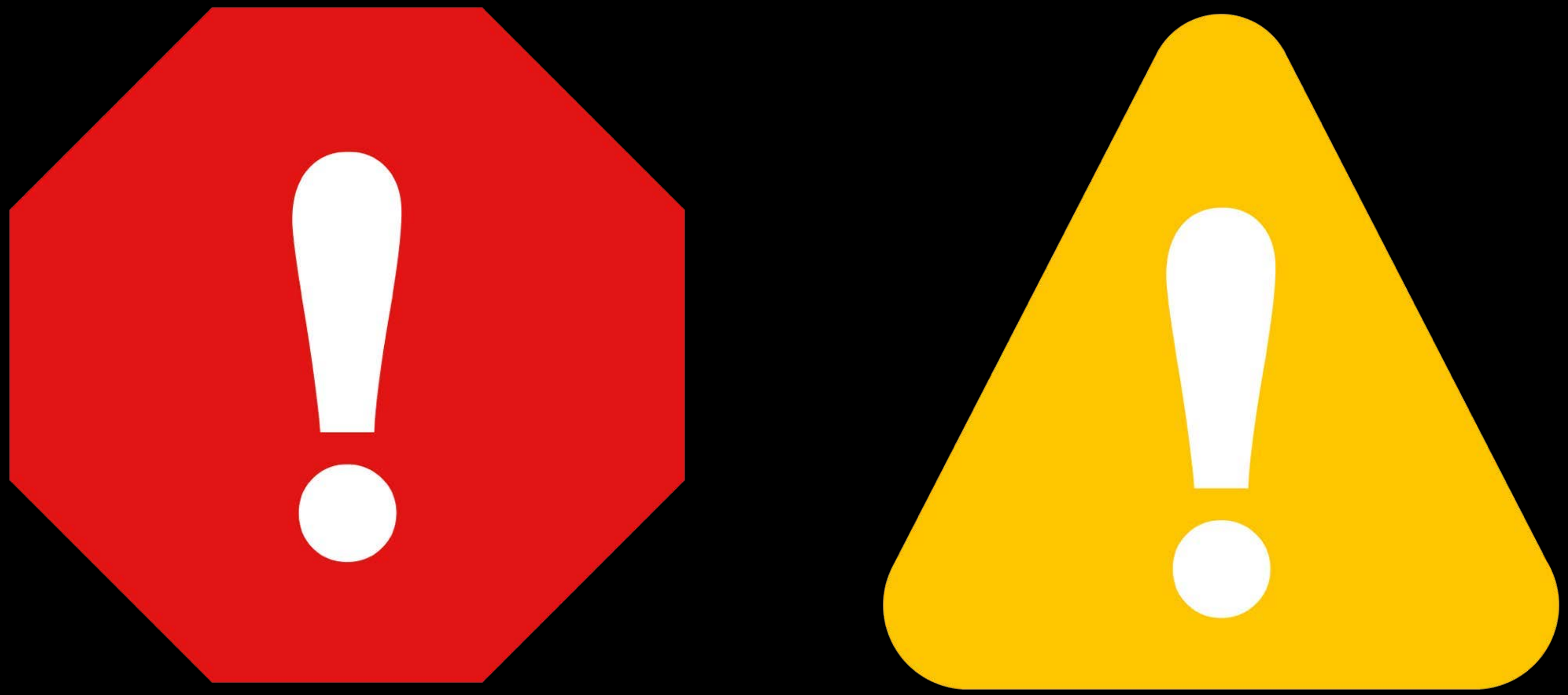
Session 406

Kuba Mracek, Program Analysis Engineer
Vedant Kumar, Compiler Engineer

Improvements in Runtime Checking

Improvements in Runtime Checking

Improvements in Runtime Checking

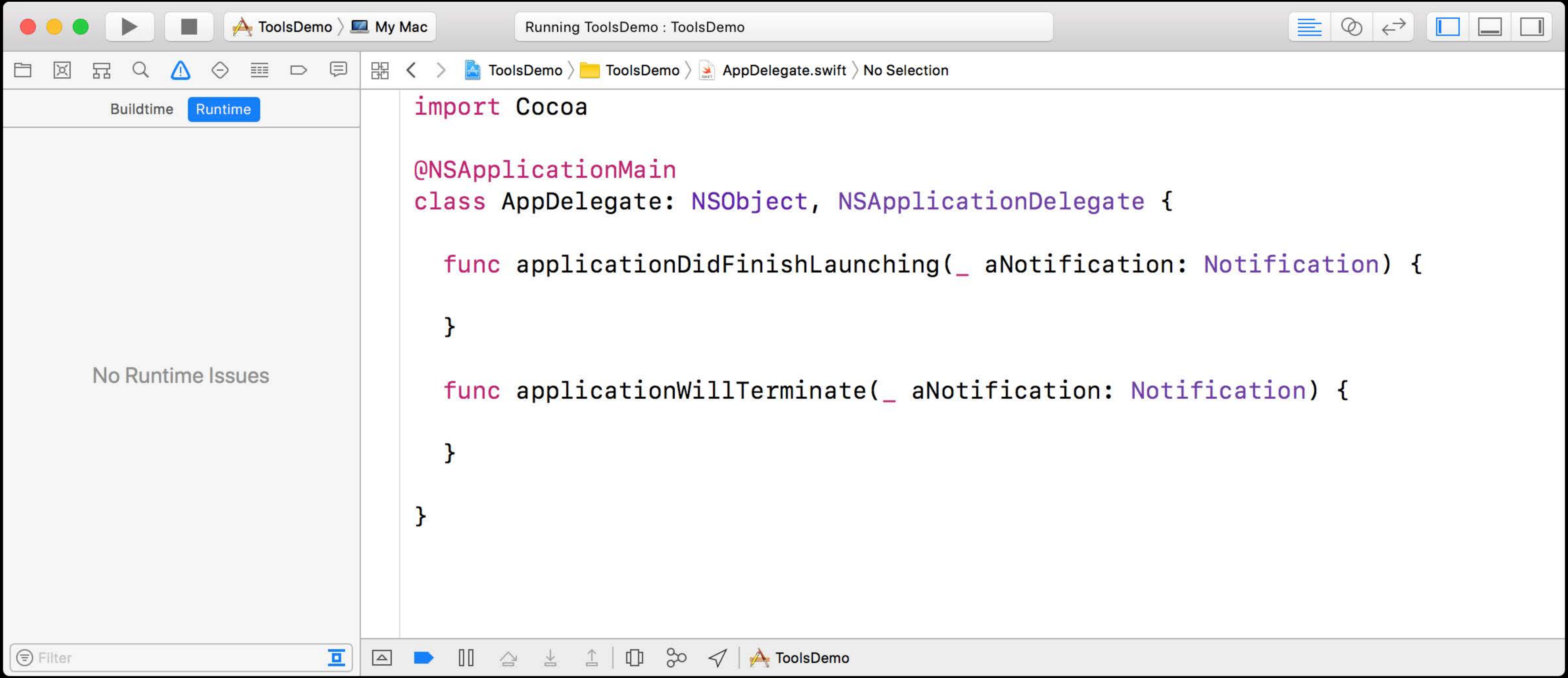


Improvements in Runtime Checking

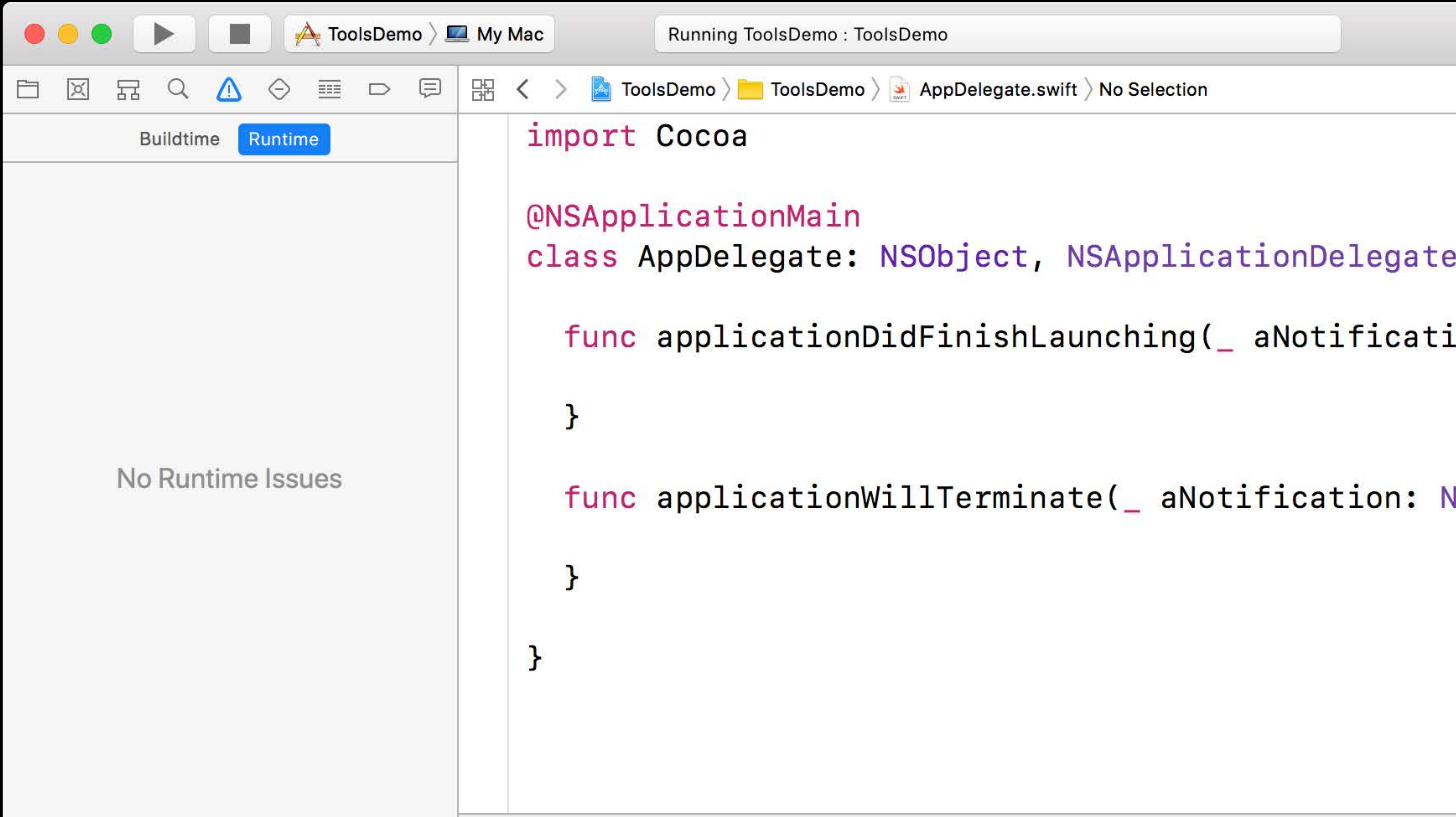
Improvements in Runtime Checking

Improvements in Runtime Checking

Runtime Issues

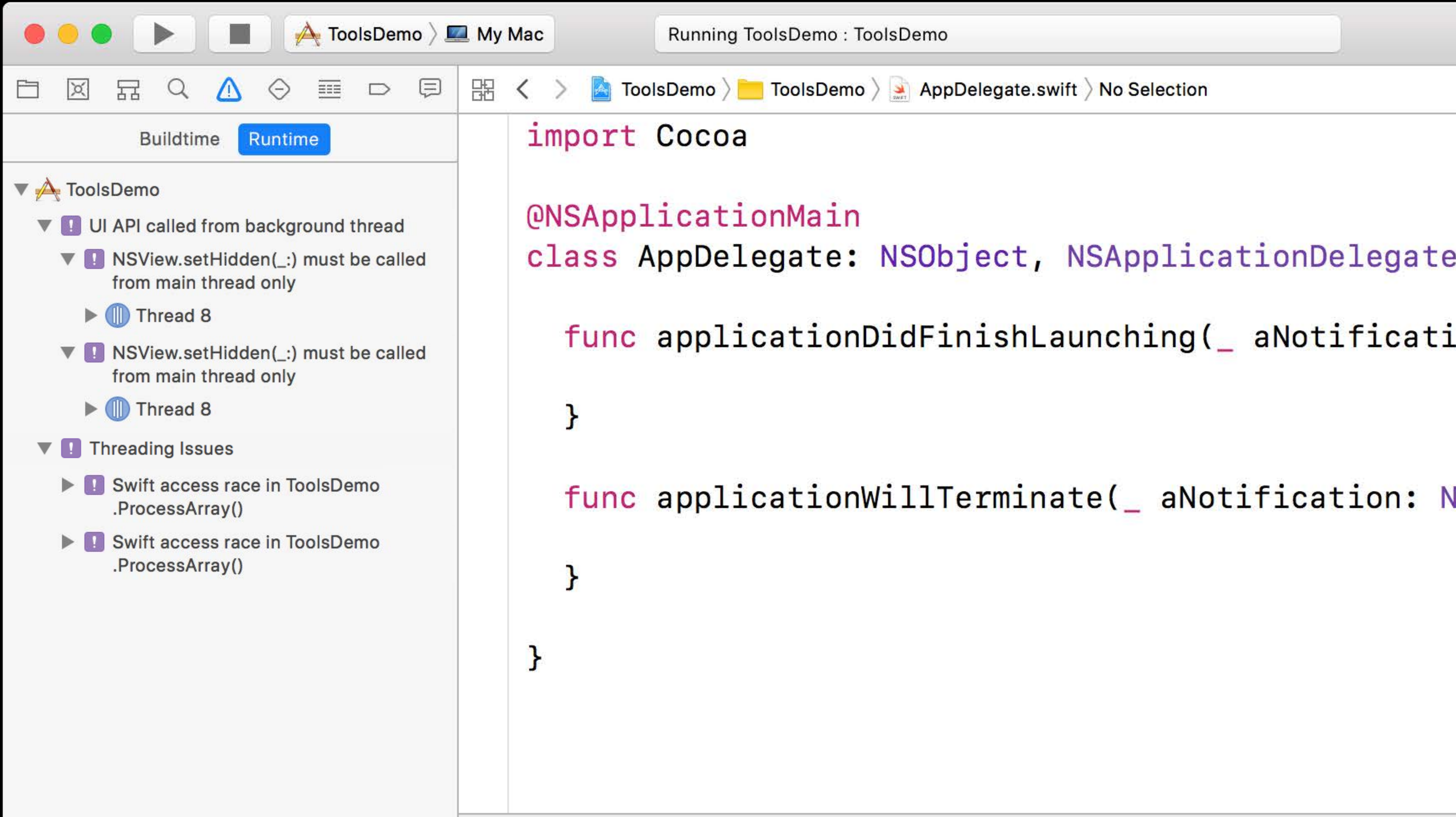


```swift
import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {

  func applicationDidFinishLaunching(_ aNotification: Notification) {

  }

  func applicationWillTerminate(_ aNotification: Notification) {

  }

}
```


ToolsDemo 〉 My Mac
Running ToolsDemo : ToolsDemo
ToolsDemo 〉 ToolsDemo 〉 AppDelegate.swift 〉 No Selection
Buildtime
Runtime

No Runtime Issues

```
import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate

    func applicationDidFinishLaunching(_ aNotificati

    }

    func applicationWillTerminate(_ aNotification: N

    }

}
```



- ToolsDemo
  - UI API called from background thread
    - NSView.setHidden(_:) must be called from main thread only
      - Thread 8
    - NSView.setHidden(_:) must be called from main thread only
      - Thread 8
  - Threading Issues
    - Swift access race in ToolsDemo .ProcessArray()
    - Swift access race in ToolsDemo .ProcessArray()

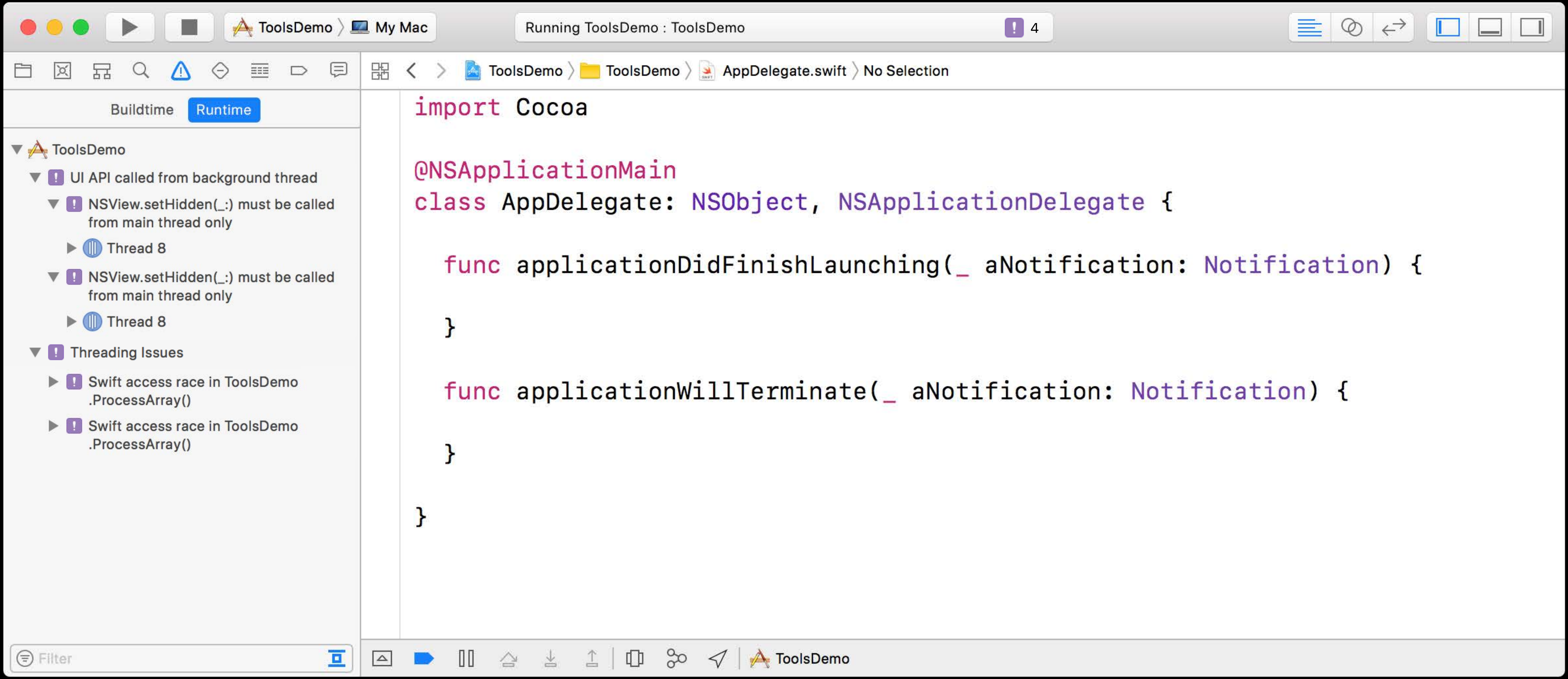
```
import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate

  func applicationDidFinishLaunching(_ aNotificati

  }

  func applicationWillTerminate(_ aNotification: N

  }

}
```

Demo
My Mac
Running ToolsDemo : ToolsDemo
ToolsDemo
ToolsDemo
AppDelegate.swift
No Selection
ead
called
called

```swift
import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {

  func applicationDidFinishLaunching(_ aNotification: Notif

  }
```

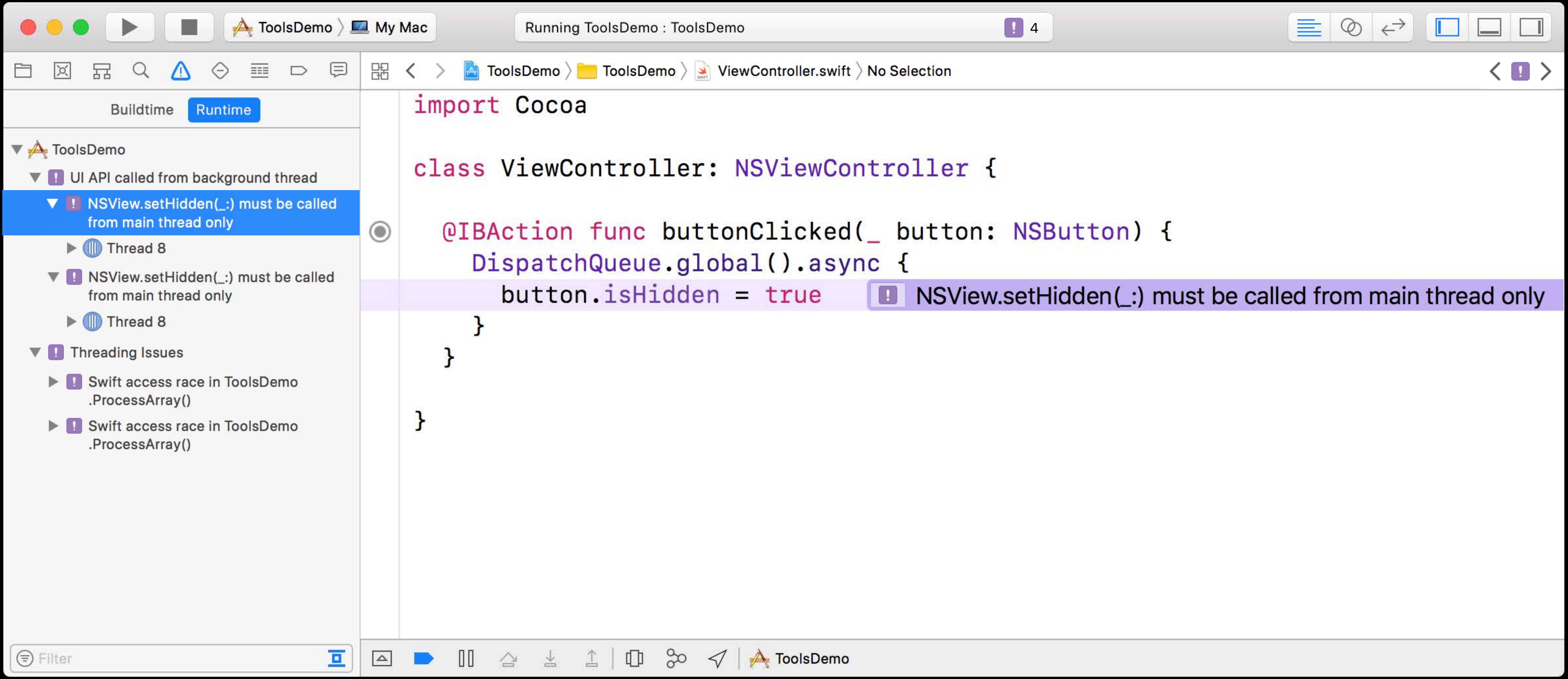
Demo
My Mac
Running ToolsDemo : ToolsDemo
4
ToolsDemo
ToolsDemo
AppDelegate.swift
No Selection
ead
called
called

```
import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {

  func applicationDidFinishLaunching(_ aNotification: Noti

  }
```



- ToolsDemo
  - UI API called from background thread
    - NSView.setHidden(_:) must be called from main thread only
      - Thread 8
    - NSView.setHidden(_:) must be called from main thread only
      - Thread 8
  - Threading Issues
    - Swift access race in ToolsDemo .ProcessArray()
    - Swift access race in ToolsDemo .ProcessArray()

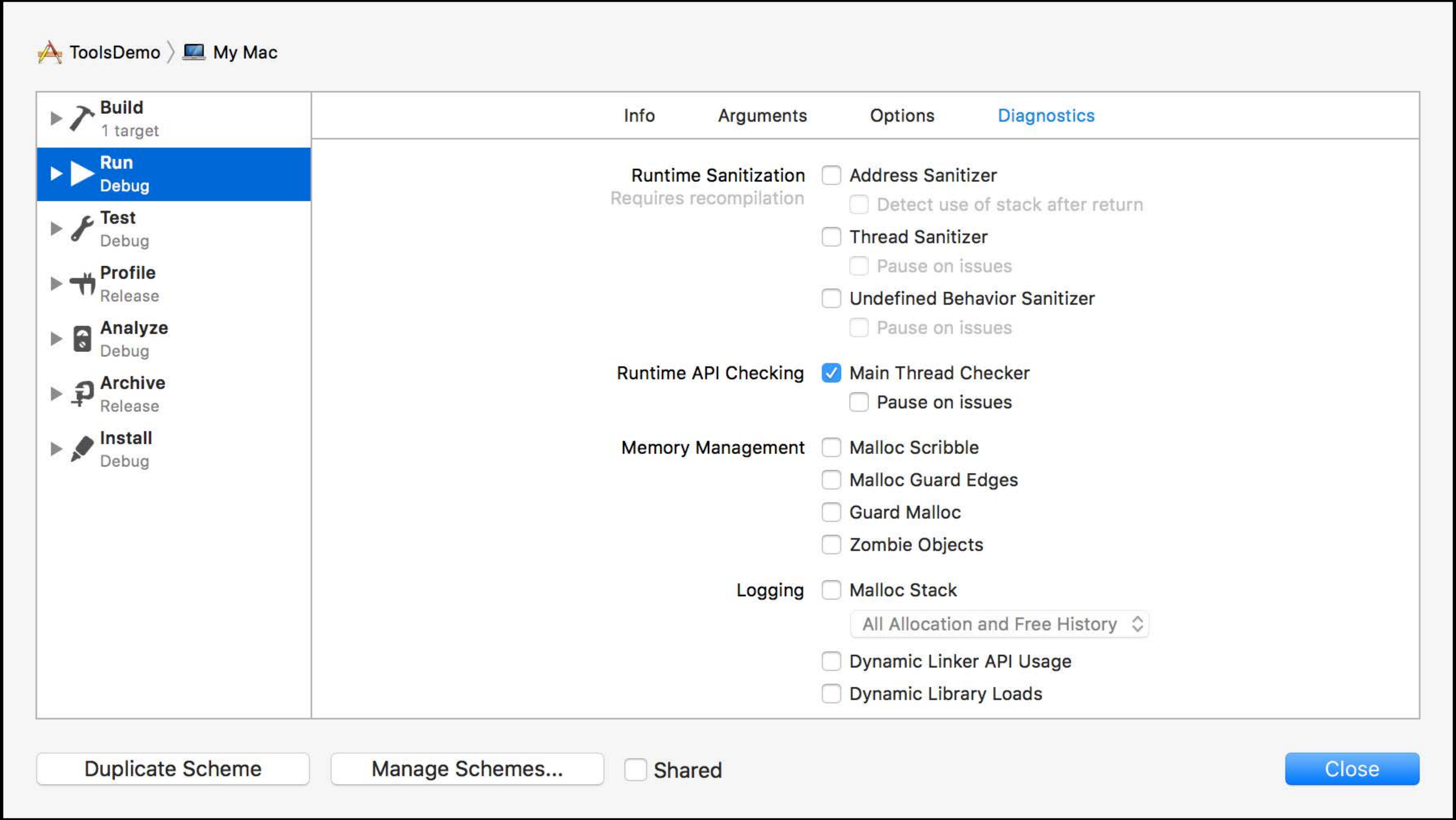
```
import Cocoa

@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {

  func applicationDidFinishLaunching(_ aNotification: Notification) {

  }

  func applicationWillTerminate(_ aNotification: Notification) {

  }

}
```


ToolsDemo 〉 My Mac
Running ToolsDemo : ToolsDemo
4
ToolsDemo 〉 ToolsDemo 〉 ViewController.swift 〉 No Selection
Buildtime
Runtime
Filter
ToolsDemo

- ToolsDemo
  - UI API called from background thread
    - NSView.setHidden(_:) must be called from main thread only
      - Thread 8
    - NSView.setHidden(_:) must be called from main thread only
      - Thread 8
  - Threading Issues
    - Swift access race in ToolsDemo .ProcessArray()
    - Swift access race in ToolsDemo .ProcessArray()

```
import Cocoa

class ViewController: NSViewController {

  @IBAction func buttonClicked(_ button: NSButton) {
    DispatchQueue.global().async {
      button.isHidden = true
    }
  }

}
```

NSView.setHidden(_:) must be called from main thread only



ToolsDemo 〉 My Mac

- ▶ Build
  1 target
- ▶ Run
  Debug
- ▶ Test
  Debug
- ▶ Profile
  Release
- ▶ Analyze
  Debug
- ▶ Archive
  Release
- ▶ Install
  Debug

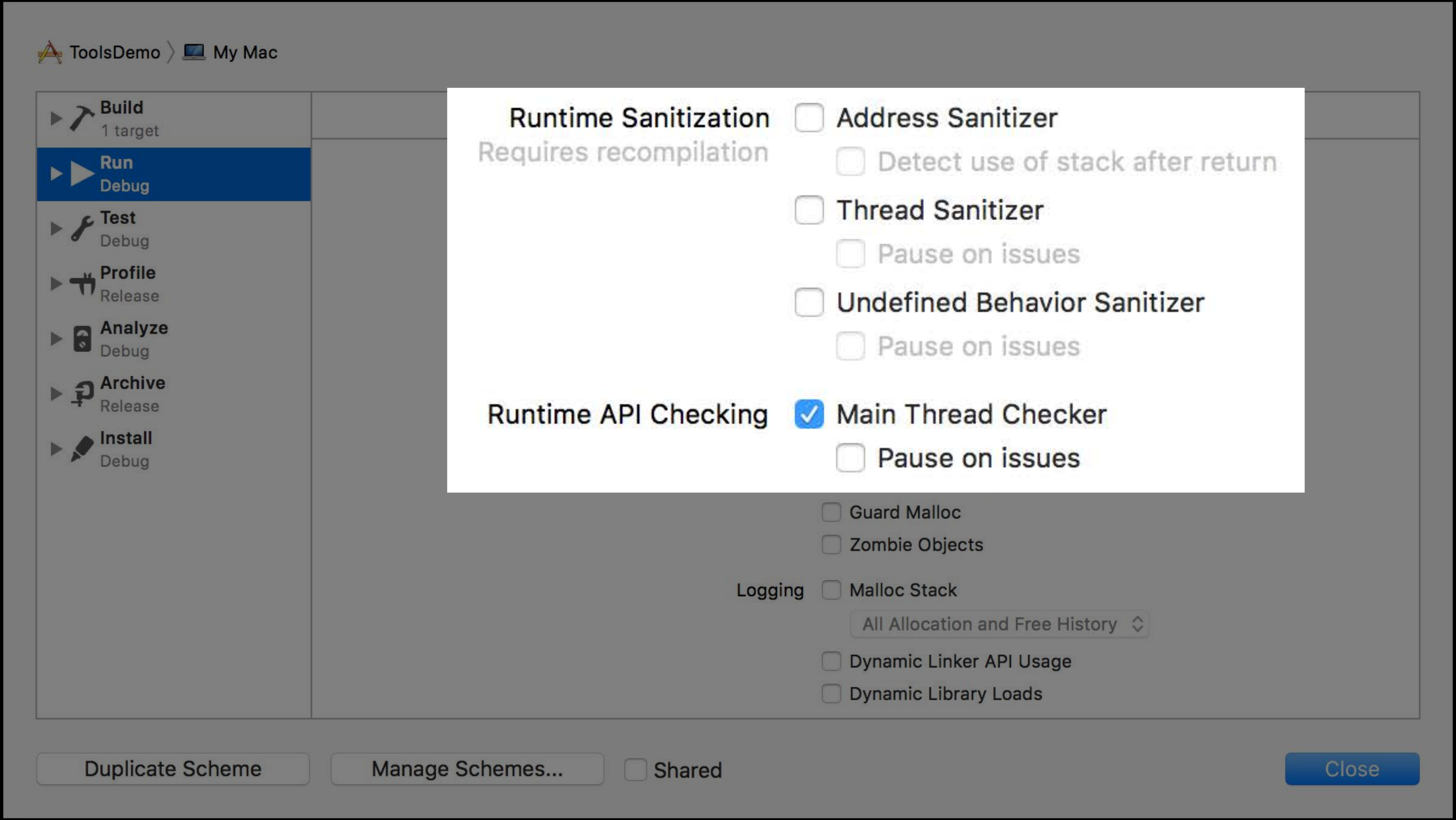
Info | Arguments | Options | Diagnostics

Runtime Sanitization
Requires recompilation
- □ Address Sanitizer
  - □ Detect use of stack after return
- □ Thread Sanitizer
  - □ Pause on issues
- □ Undefined Behavior Sanitizer
  - □ Pause on issues

Runtime API Checking
- ☑ Main Thread Checker
  - □ Pause on issues

Memory Management
- □ Malloc Scribble
- □ Malloc Guard Edges
- □ Guard Malloc
- □ Zombie Objects

Logging
- □ Malloc Stack
  - All Allocation and Free History ⌃⌄
- □ Dynamic Linker API Usage
- □ Dynamic Library Loads



Runtime Sanitization
Requires recompilation

- ☐ Address Sanitizer
  - ☐ Detect use of stack after return
- ☐ Thread Sanitizer
  - ☐ Pause on issues
- ☐ Undefined Behavior Sanitizer
  - ☐ Pause on issues

Runtime API Checking

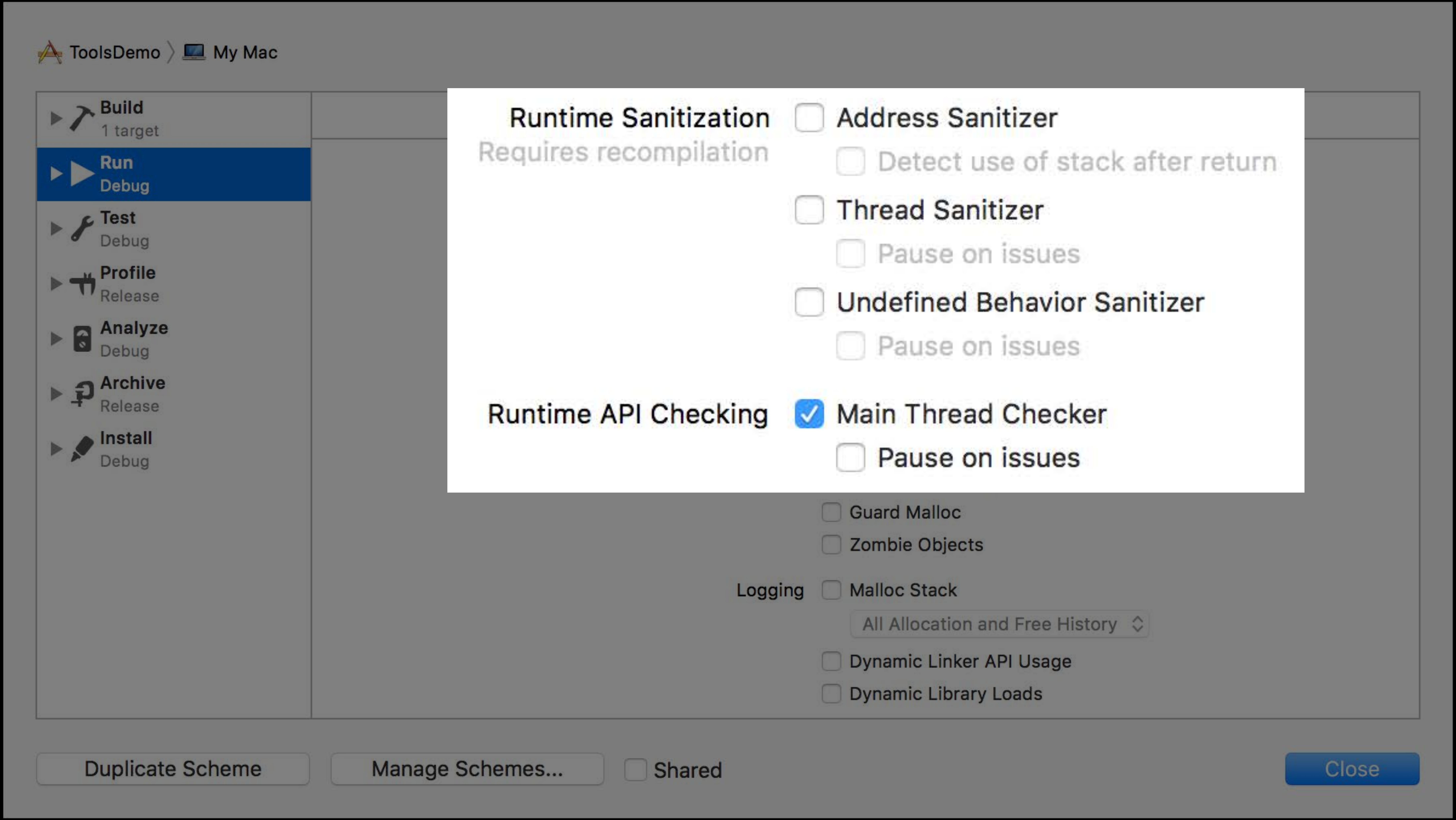
- ☑ Main Thread Checker
  - ☐ Pause on issues

- ☐ Guard Malloc
- ☐ Zombie Objects

Logging

- ☐ Malloc Stack
  - All Allocation and Free History ⇕
- ☐ Dynamic Linker API Usage
- ☐ Dynamic Library Loads

☐ Shared



Runtime Sanitization
Requires recompilation

- □ Address Sanitizer
  - □ Detect use of stack after return
- □ Thread Sanitizer
  - □ Pause on issues
- □ Undefined Behavior Sanitizer
  - □ Pause on issues

Runtime API Checking

- ☑ Main Thread Checker
  - □ Pause on issues

- □ Guard Malloc
- □ Zombie Objects

Logging

- □ Malloc Stack
  - All Allocation and Free History
- □ Dynamic Linker API Usage
- □ Dynamic Library Loads

□ Shared

Address Sanitizer

Thread Sanitizer

Undefined Behavior Sanitizer

Main Thread Checker

Main Thread Checker

Address Sanitizer

Thread Sanitizer

Undefined Behavior Sanitizer

Using Runtime Tools Effectively

Main Thread Checker New

Address Sanitizer

Thread Sanitizer

Undefined Behavior Sanitizer

Using Runtime Tools Effectively

Main Thread Checker	New
Address Sanitizer	
Thread Sanitizer	
Undefined Behavior Sanitizer	New
Using Runtime Tools Effectively	

NEW

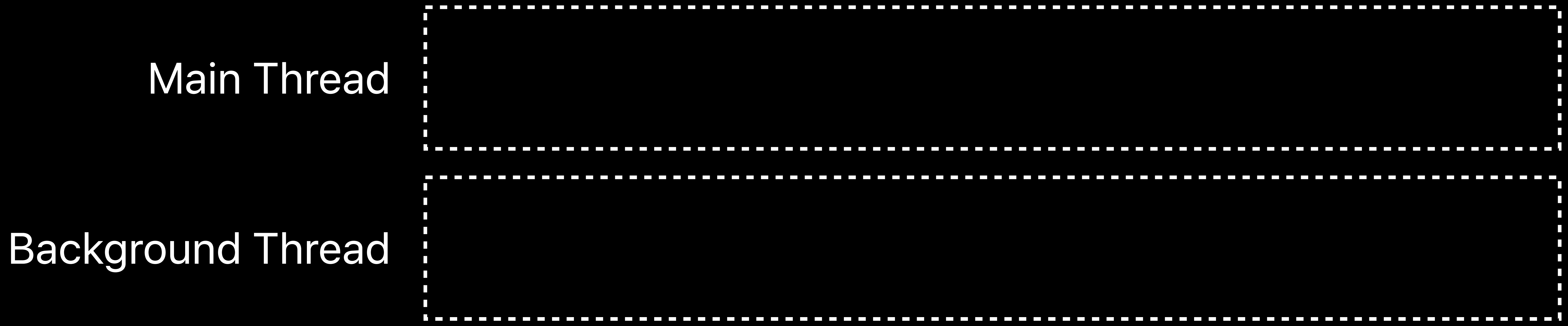
Main Thread Checker

Detects misuses of common APIs

UI Updates and Threads

Some APIs must only be used from the main thread

UI Updates and Threads



Main Thread

Background Thread

UI Updates and Threads

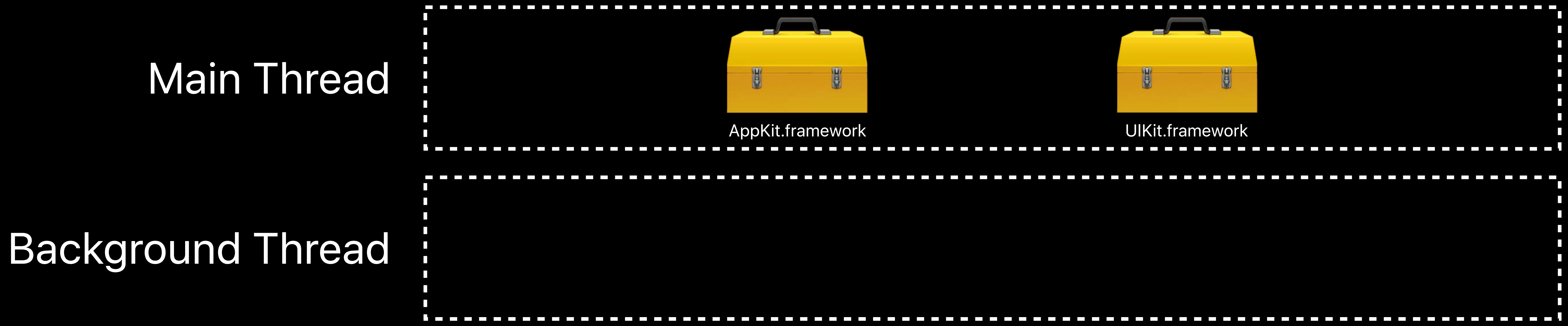
Main Thread

Background Thread

AppKit.framework

UIKit.framework

UI Updates and Threads



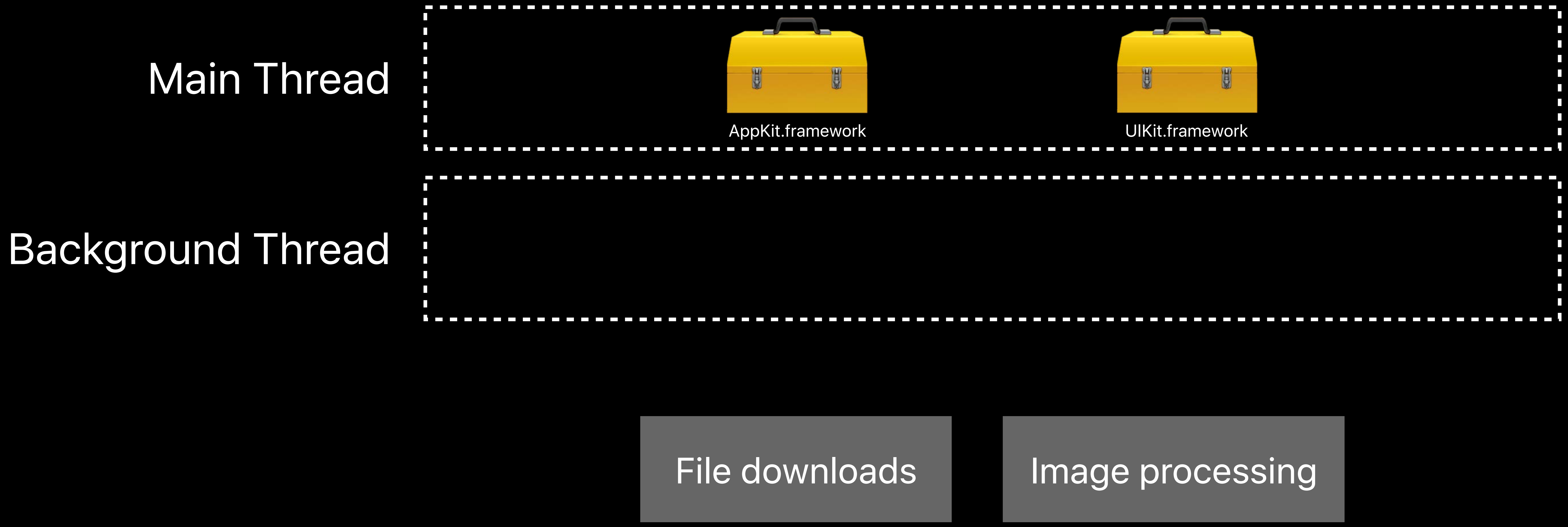
Main Thread

AppKit.framework

UIKit.framework

Background Thread

UI Updates and Threads



Main Thread

AppKit.framework

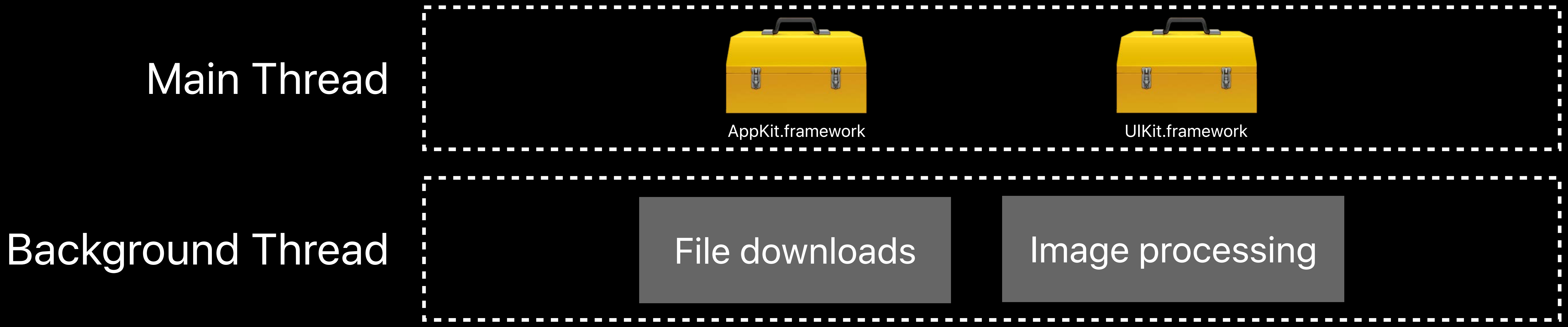
UIKit.framework

Background Thread

File downloads

Image processing

UI Updates and Threads



Main Thread

AppKit.framework

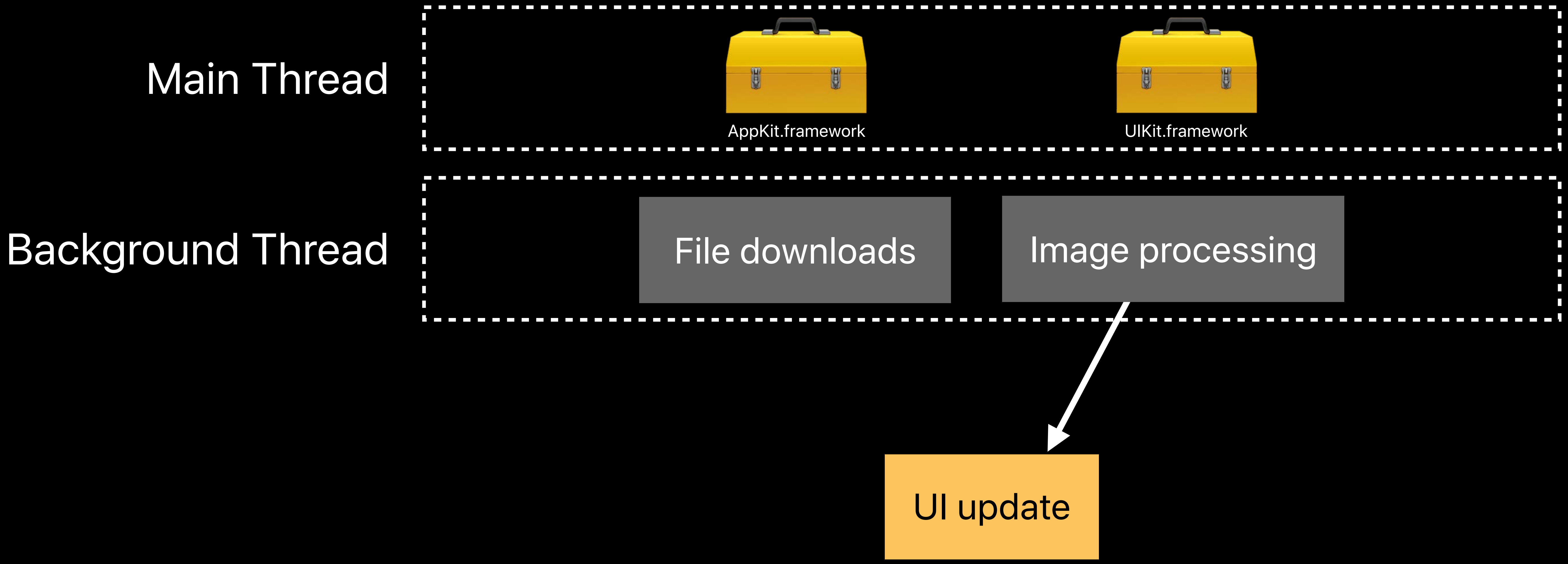
UIKit.framework

Background Thread

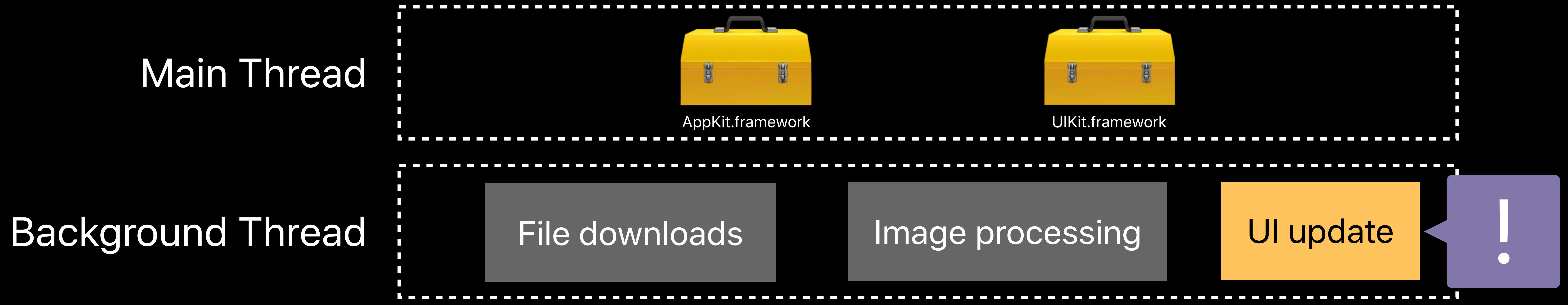
File downloads

Image processing

UI Updates and Threads



UI Updates and Threads



Main Thread

AppKit.framework

UIKit.framework

Background Thread

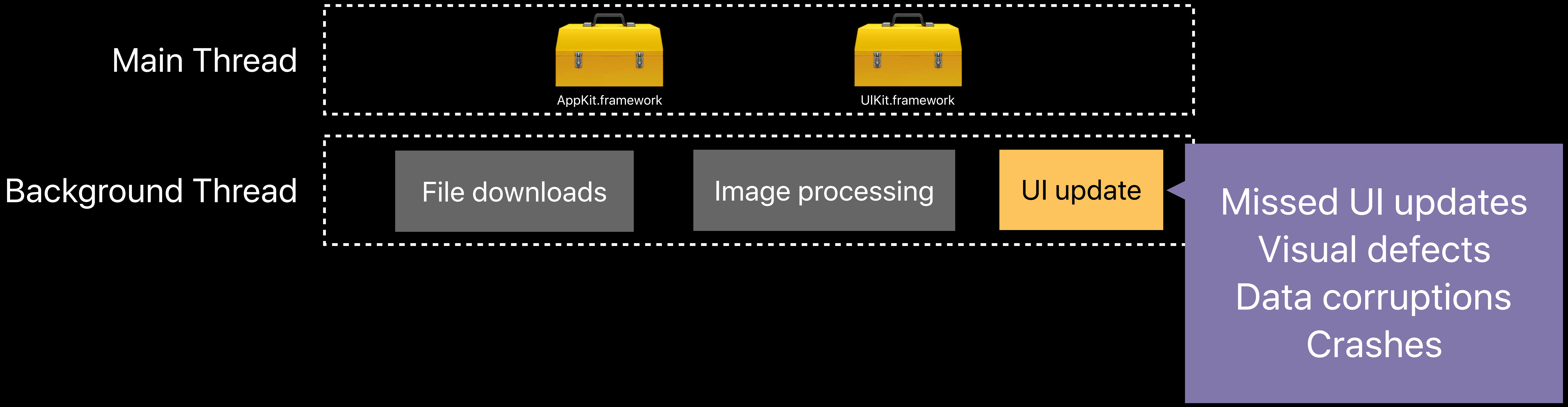
File downloads

Image processing

UI update

!

UI Updates and Threads



Main Thread

AppKit.framework

UIKit.framework

Background Thread

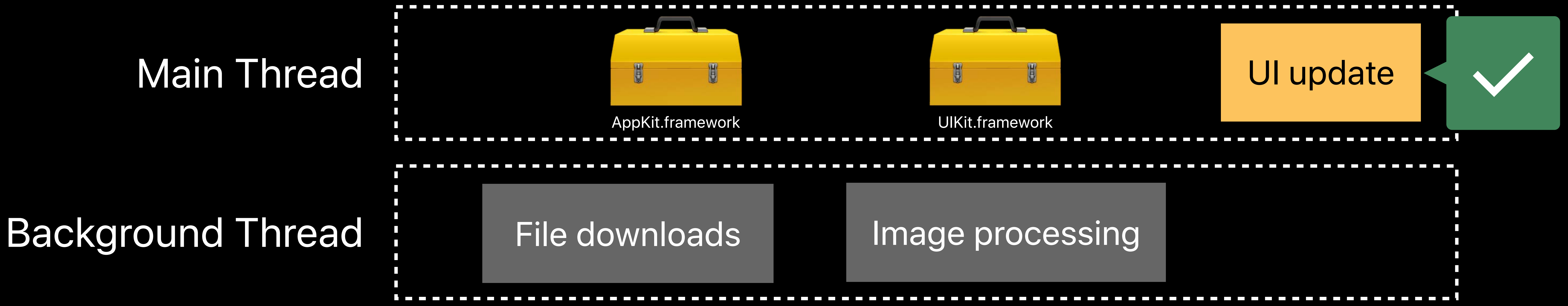
File downloads

Image processing

UI update

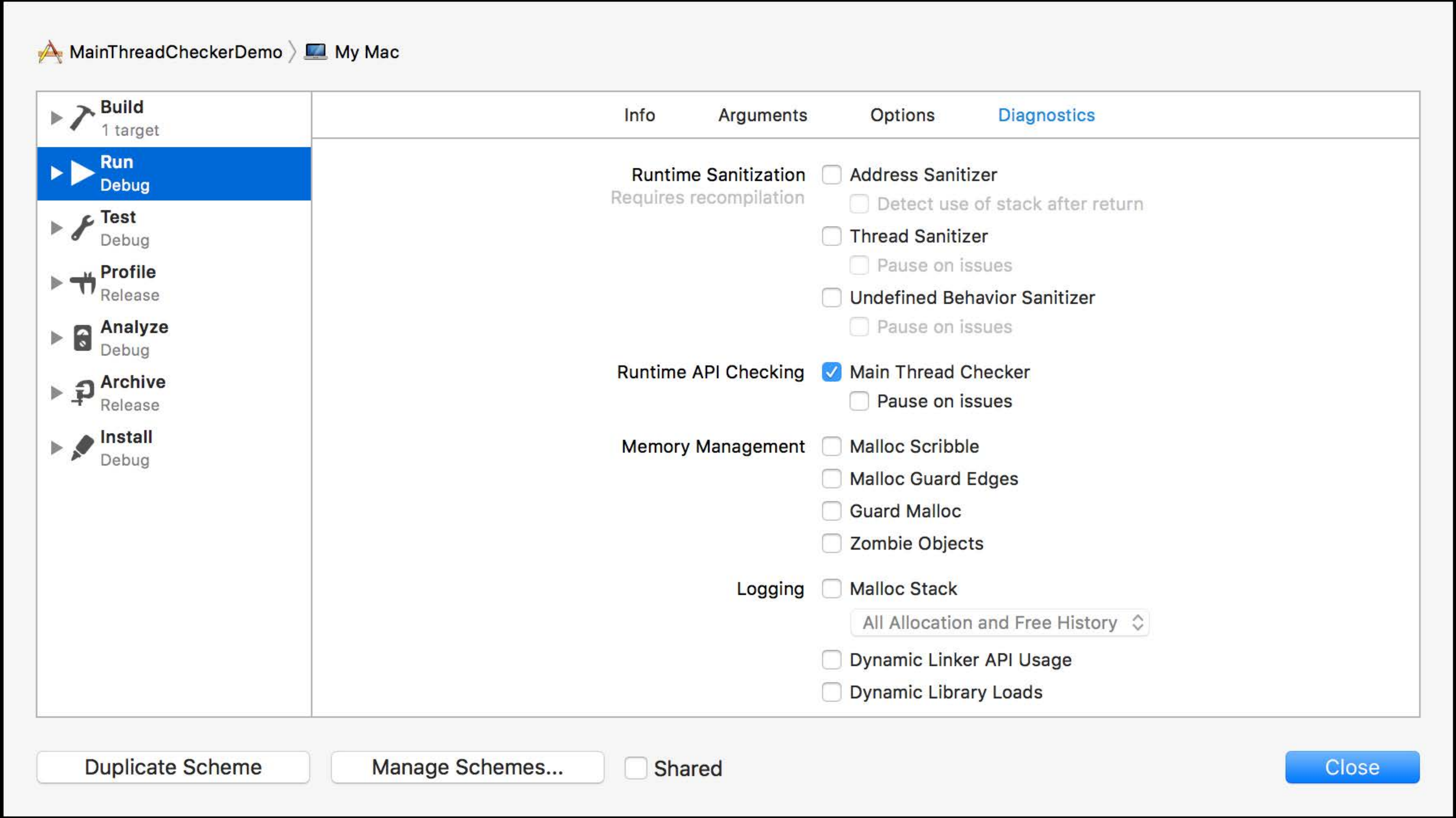
Missed UI updates
Visual defects
Data corruptions
Crashes

UI Updates and Threads



Demo

Main Thread Checker



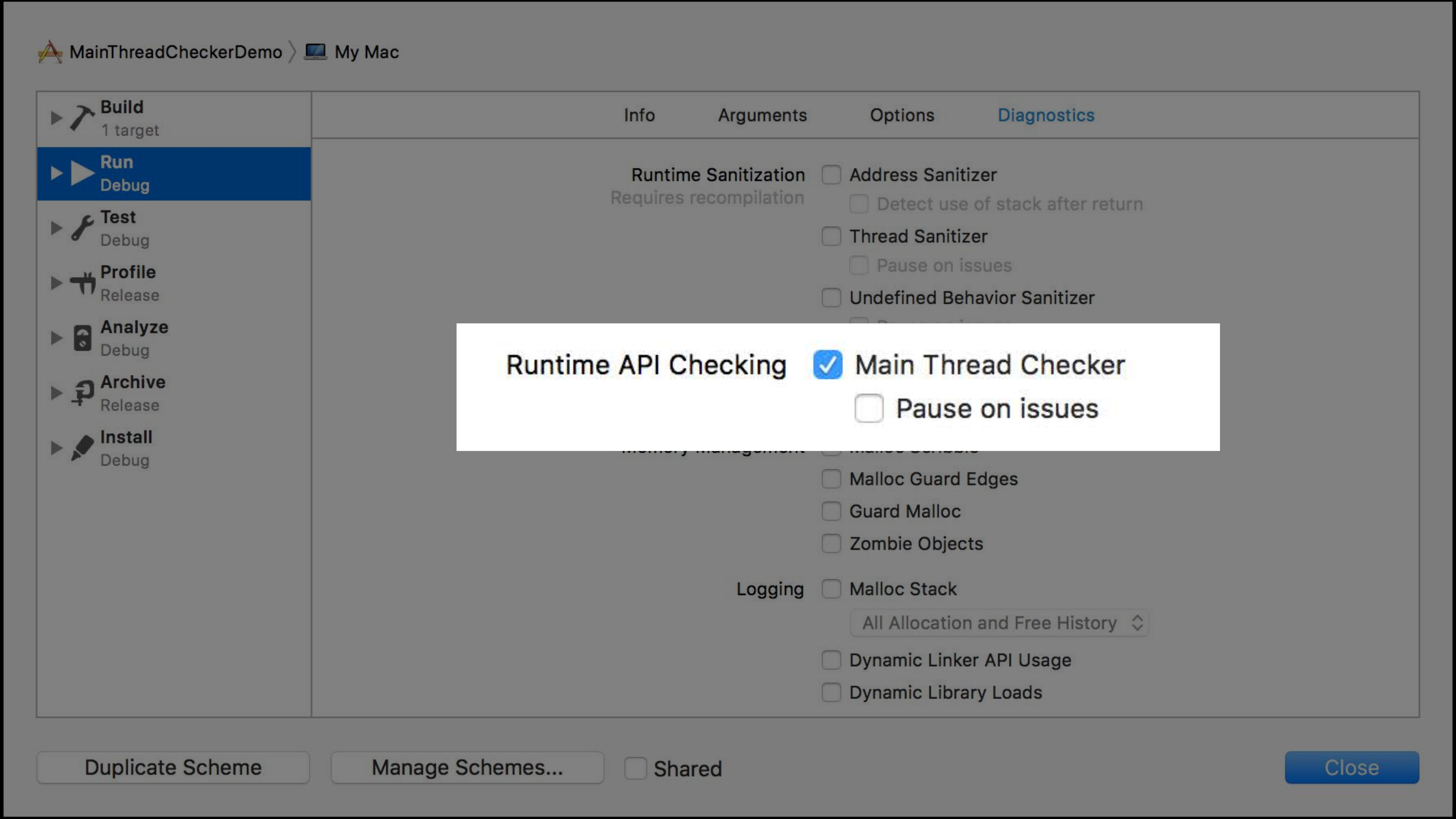
Runtime Sanitization
Requires recompilation
- □ Address Sanitizer
  - □ Detect use of stack after return
- □ Thread Sanitizer
  - □ Pause on issues
- □ Undefined Behavior Sanitizer
  - □ Pause on issues

Runtime API Checking
- ☑ Main Thread Checker
  - □ Pause on issues

Memory Management
- □ Malloc Scribble
- □ Malloc Guard Edges
- □ Guard Malloc
- □ Zombie Objects

Logging
- □ Malloc Stack
  - All Allocation and Free History
- □ Dynamic Linker API Usage
- □ Dynamic Library Loads

□ Shared



MainThreadCheckerDemo 〉 My Mac

- Build — 1 target
- Run — Debug
- Test — Debug
- Profile — Release
- Analyze — Debug
- Archive — Release
- Install — Debug

Info | Arguments | Options | Diagnostics

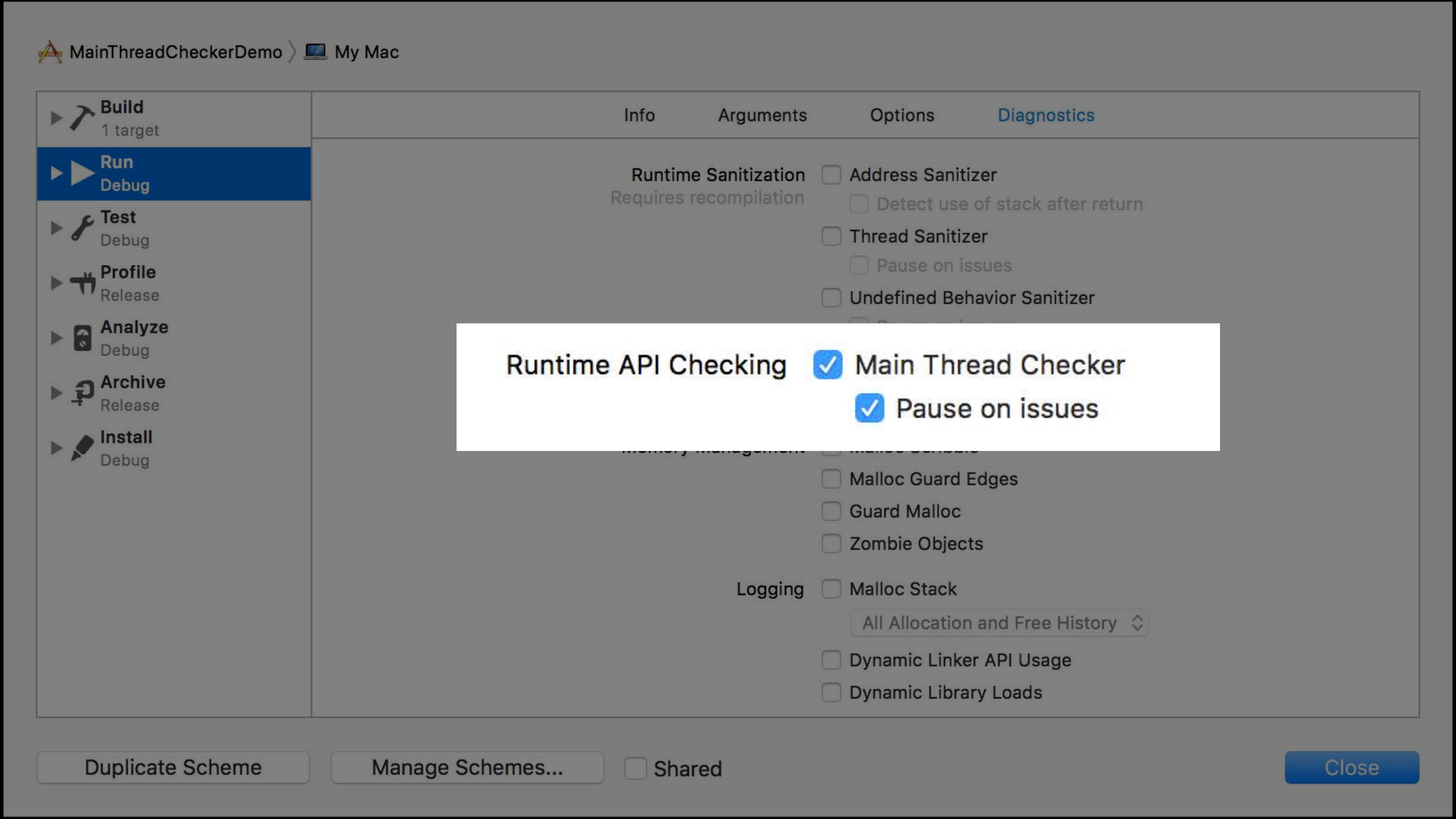
Runtime Sanitization
Requires recompilation

□ Address Sanitizer
□ Detect use of stack after return
□ Thread Sanitizer
□ Pause on issues
□ Undefined Behavior Sanitizer

Runtime API Checking ☑ Main Thread Checker
□ Pause on issues

Memory Management
□ Malloc Guard Edges
□ Guard Malloc
□ Zombie Objects

Logging □ Malloc Stack
All Allocation and Free History
□ Dynamic Linker API Usage
□ Dynamic Library Loads



MainThreadCheckerDemo 〉 My Mac

Runtime Sanitization
Requires recompilation
☐ Address Sanitizer
☐ Detect use of stack after return
☐ Thread Sanitizer
☐ Pause on issues
☐ Undefined Behavior Sanitizer

Runtime API Checking
☑ Main Thread Checker
☑ Pause on issues

☐ Malloc Guard Edges
☐ Guard Malloc
☐ Zombie Objects

Logging
☐ Malloc Stack
All Allocation and Free History
☐ Dynamic Linker API Usage
☐ Dynamic Library Loads

☐ Shared

Common Places for Mistakes

Networking callbacks

Creating and destroying NSView and UIView objects

Designing asynchronous APIs

Designing Asynchronous APIs

Let API user specify callback queue

Designing Asynchronous APIs

Let API user specify callback queue

```
DeepThought.asyncComputeAnswer(to: theQuestion) { reply in
   …
}
```

✗

Designing Asynchronous APIs

Let API user specify callback queue

```
DeepThought.asyncComputeAnswer(to: theQuestion, completionQueue: queue) { reply in
   …
}
```

✓

Main Thread Checker

NEW

Detects violations of API threading rules

AppKit, UIKit and WebKit APIs

Swift and C languages

No recompilation

Enabled by default in the Xcode debugger

Address Sanitizer

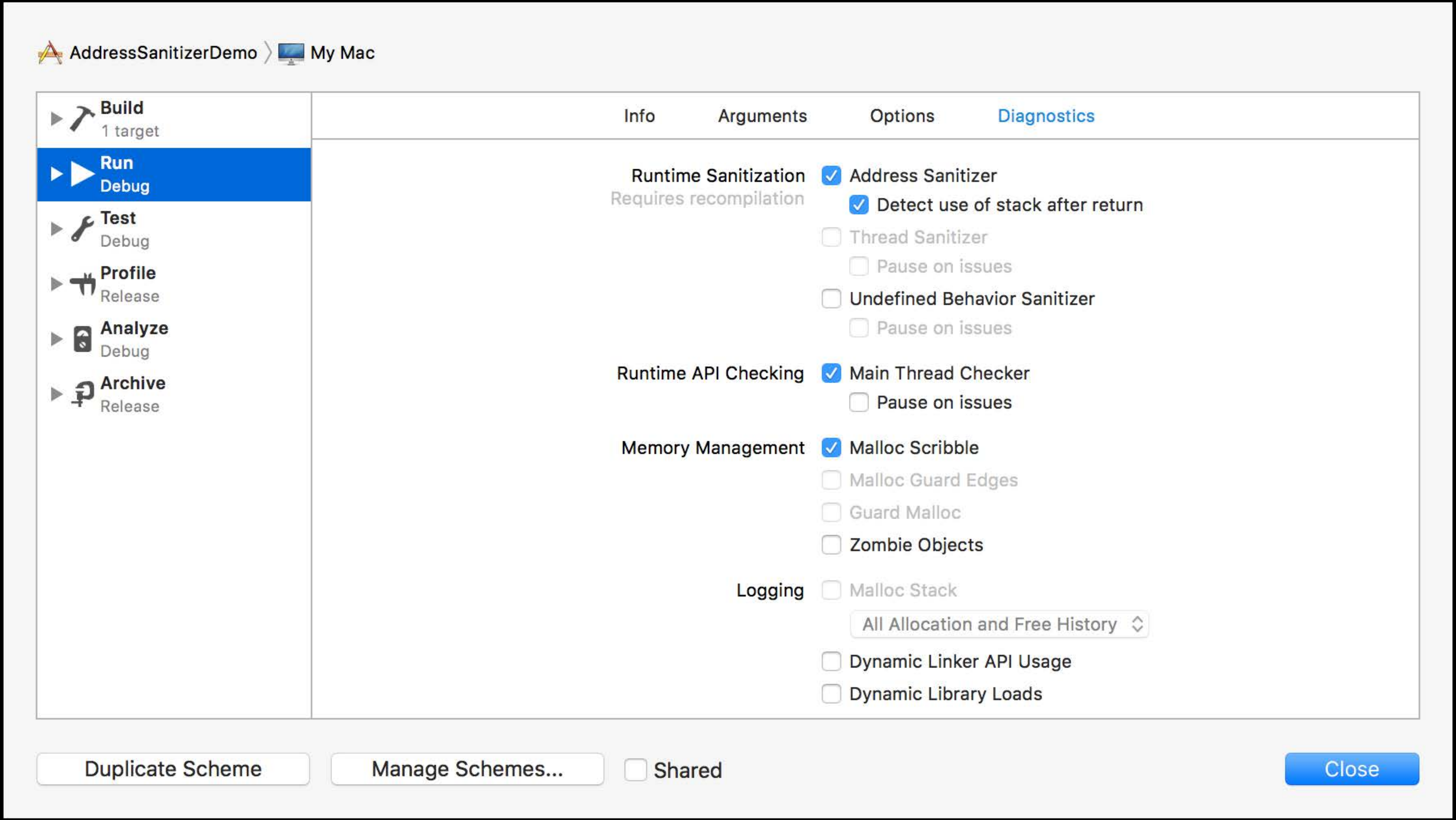
Detects memory issues

Finding Memory Issues

Security critical bugs

- Use-after-free and buffer overflows

Diagnoses hard-to-reproduce crashes



Runtime Sanitization
Requires recompilation

☑ Address Sanitizer
☑ Detect use of stack after return
□ Thread Sanitizer
□ Pause on issues
□ Undefined Behavior Sanitizer
□ Pause on issues

Runtime API Checking

☑ Main Thread Checker
□ Pause on issues

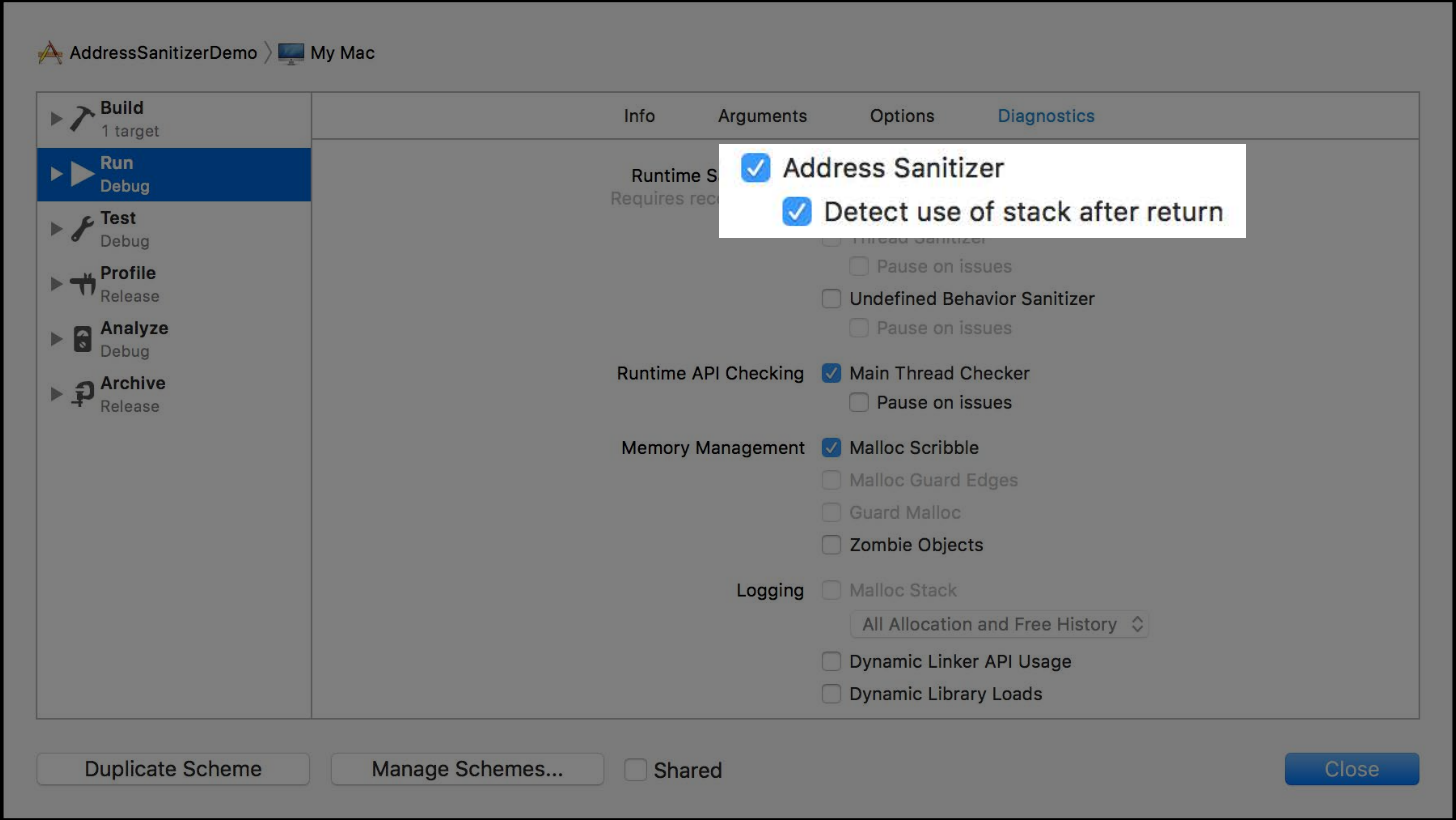
Memory Management

☑ Malloc Scribble
□ Malloc Guard Edges
□ Guard Malloc
□ Zombie Objects

Logging

□ Malloc Stack
All Allocation and Free History
□ Dynamic Linker API Usage
□ Dynamic Library Loads

□ Shared



Runtime S
Requires rec

☑ Address Sanitizer

☑ Detect use of stack after return

□ Thread Sanitizer

□ Pause on issues

□ Undefined Behavior Sanitizer

□ Pause on issues

Runtime API Checking ☑ Main Thread Checker

□ Pause on issues

Memory Management ☑ Malloc Scribble

□ Malloc Guard Edges

□ Guard Malloc

□ Zombie Objects

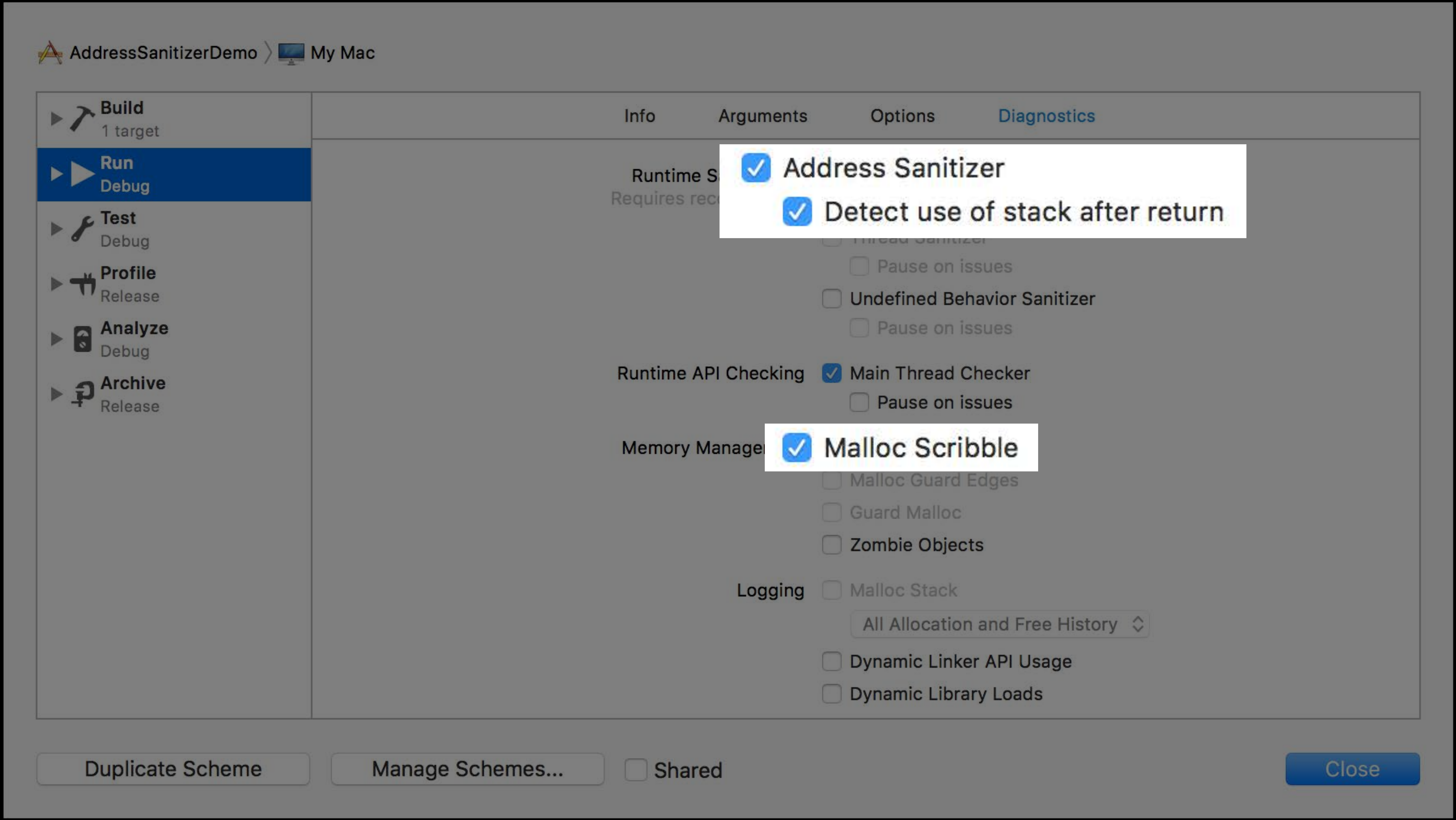
Logging □ Malloc Stack

All Allocation and Free History

□ Dynamic Linker API Usage

□ Dynamic Library Loads

□ Shared



Runtime S

Requires rec

☑ Address Sanitizer

☑ Detect use of stack after return

☐ Thread Sanitizer

☐ Pause on issues

☐ Undefined Behavior Sanitizer

☐ Pause on issues

Runtime API Checking ☑ Main Thread Checker

☐ Pause on issues

Memory Manage ☑ Malloc Scribble

☐ Malloc Guard Edges

☐ Guard Malloc

☐ Zombie Objects

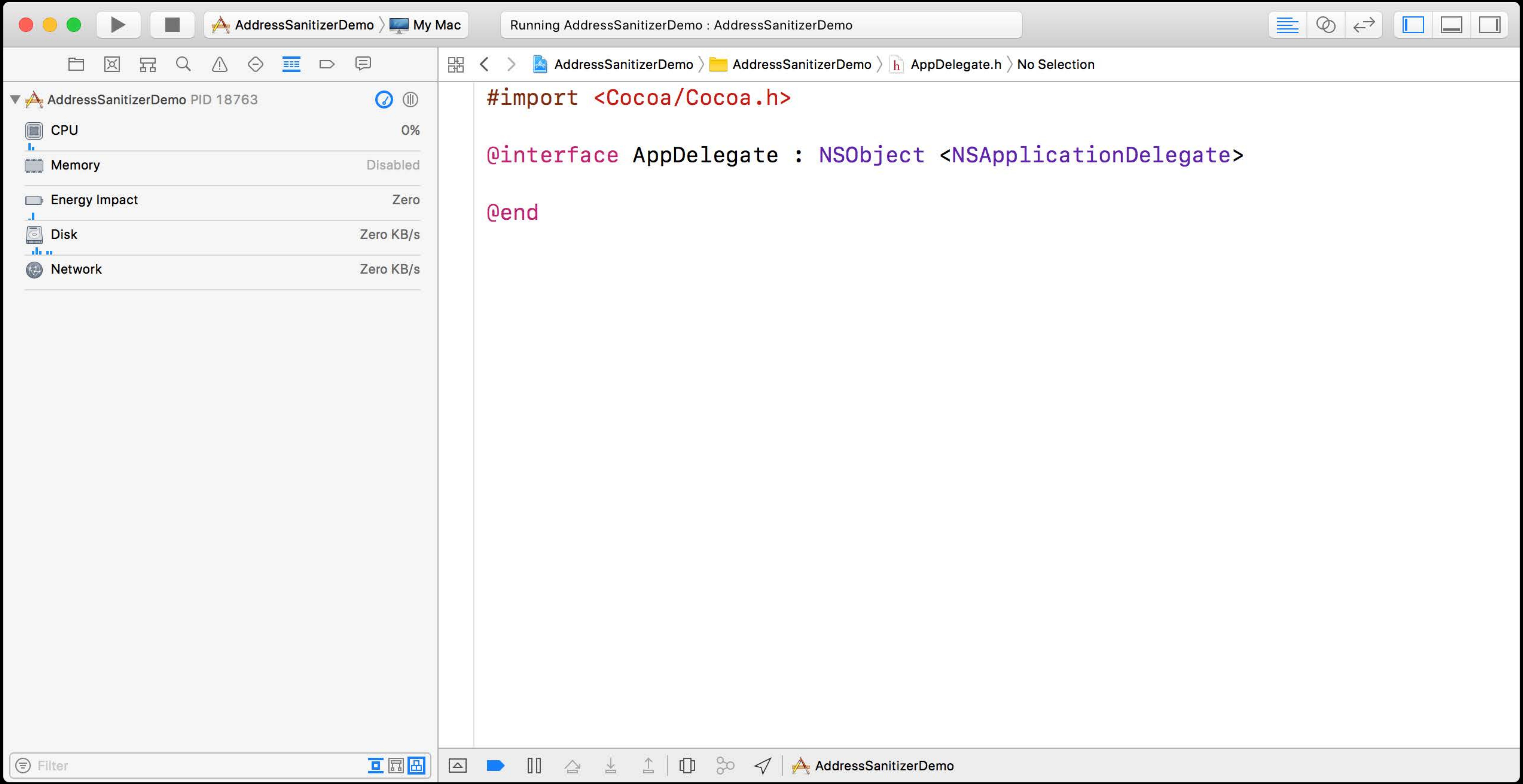
Logging ☐ Malloc Stack

All Allocation and Free History

☐ Dynamic Linker API Usage

☐ Dynamic Library Loads

☐ Shared

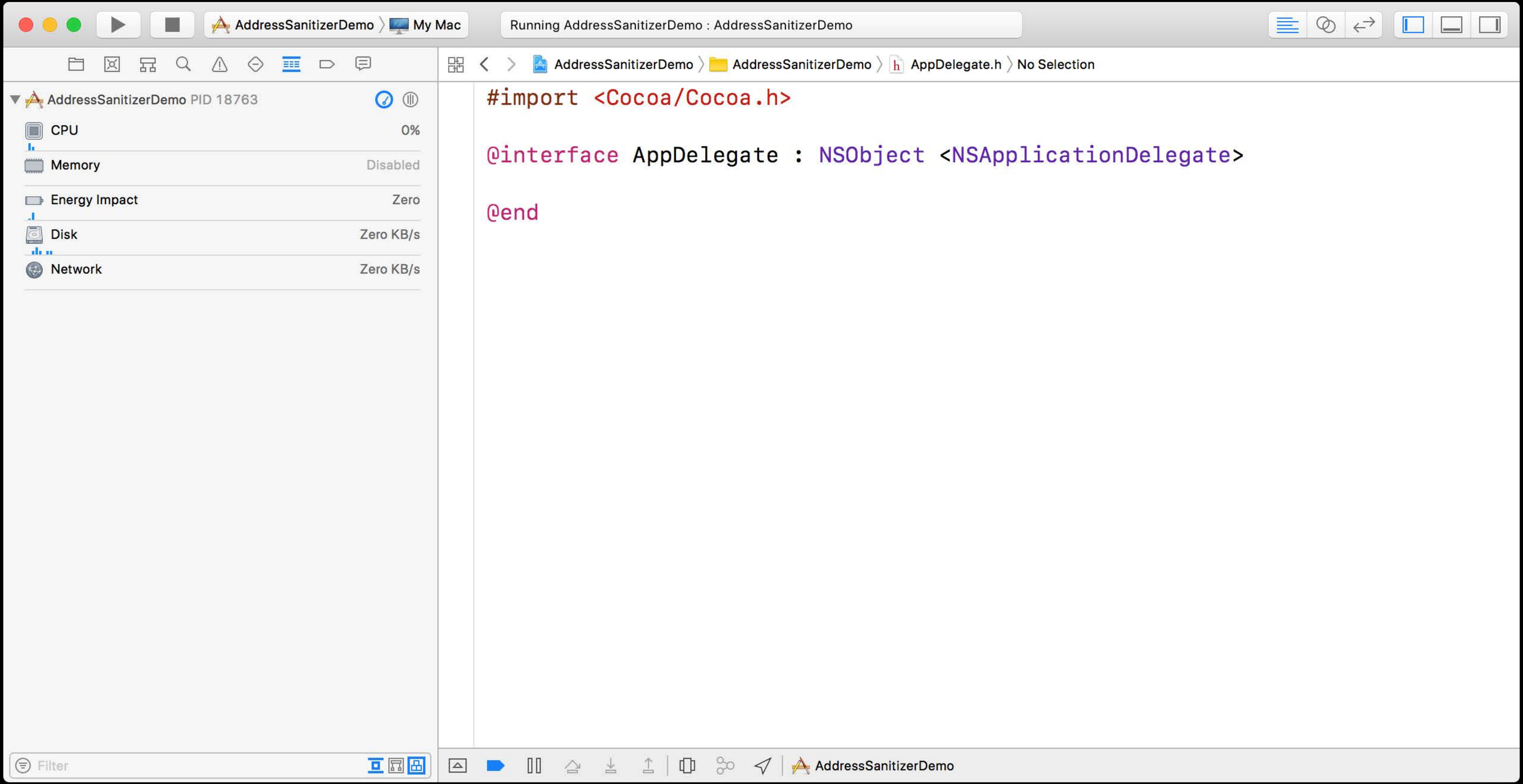


Running AddressSanitizerDemo : AddressSanitizerDemo

AddressSanitizerDemo PID 18763

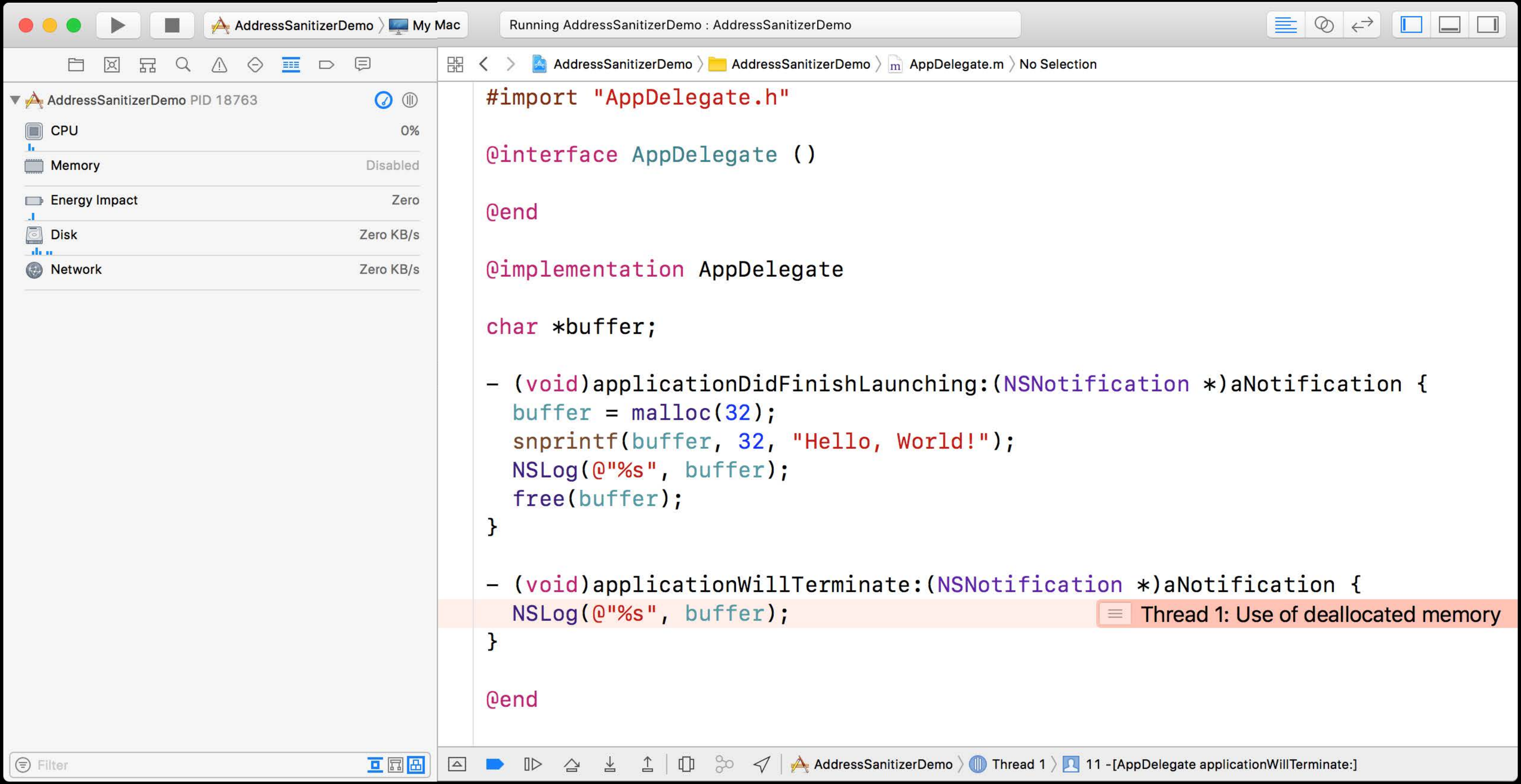
```
CPU              0%
Memory           Disabled
Energy Impact    Zero
Disk             Zero KB/s
Network          Zero KB/s
```

```
#import <Cocoa/Cocoa.h>

@interface AppDelegate : NSObject <NSApplicationDelegate>

@end
```



AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 AddressSanitizerDemo 〉 AppDelegate.h 〉 No Selection
AddressSanitizerDemo PID 18763
CPU 0%
Memory Disabled
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Filter
AddressSanitizerDemo

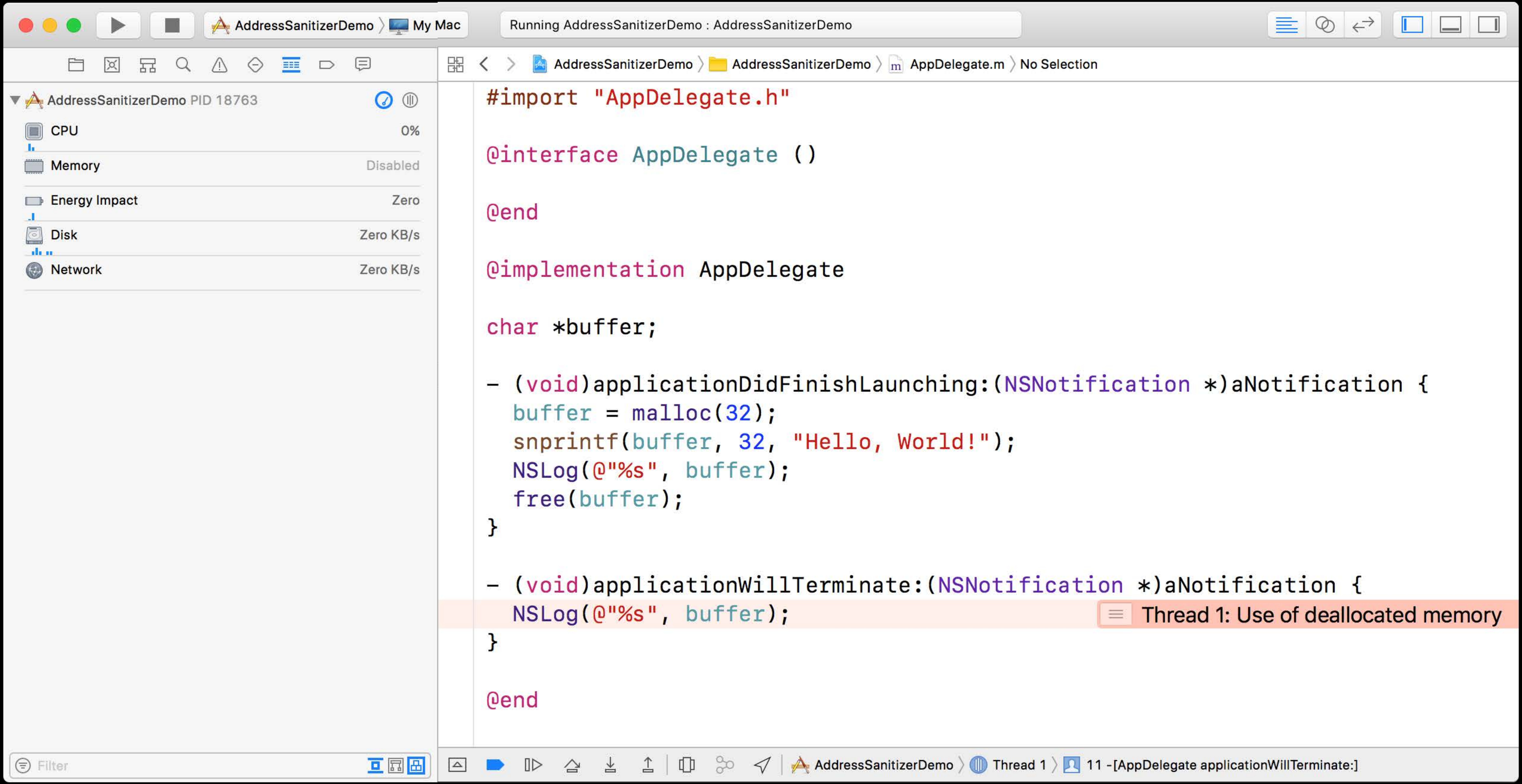
```
#import <Cocoa/Cocoa.h>

@interface AppDelegate : NSObject <NSApplicationDelegate>

@end
```



AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 AddressSanitizerDemo 〉 AppDelegate.m 〉 No Selection
AddressSanitizerDemo PID 18763
CPU 0%
Memory Disabled
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Filter
AddressSanitizerDemo 〉 Thread 1 〉 11 -[AppDelegate applicationWillTerminate:]

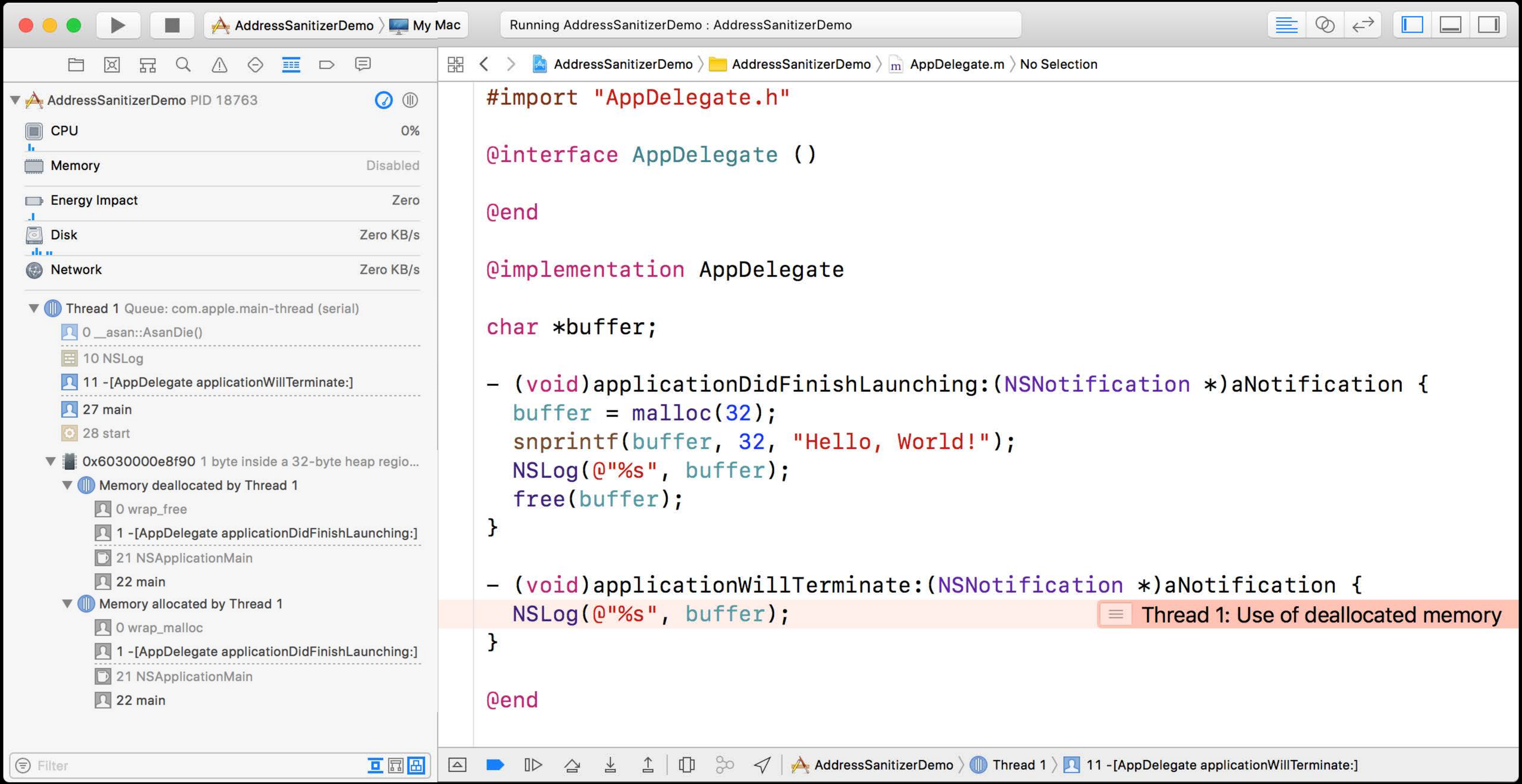
```
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

char *buffer;

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
  buffer = malloc(32);
  snprintf(buffer, 32, "Hello, World!");
  NSLog(@"%s", buffer);
  free(buffer);
}

- (void)applicationWillTerminate:(NSNotification *)aNotification {
  NSLog(@"%s", buffer);
}

@end
```

Thread 1: Use of deallocated memory



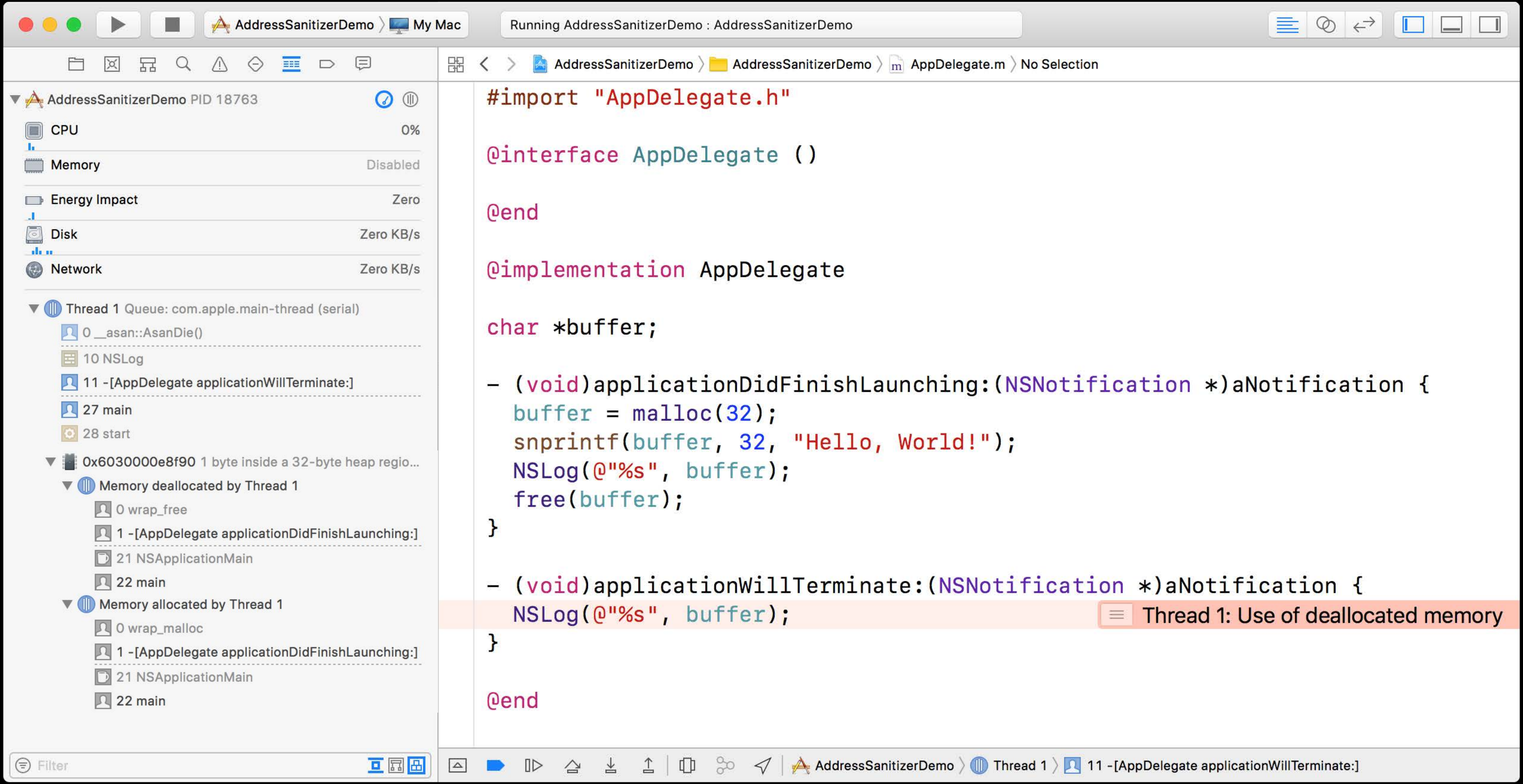
```
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

char *buffer;

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
  buffer = malloc(32);
  snprintf(buffer, 32, "Hello, World!");
  NSLog(@"%s", buffer);
  free(buffer);
}

- (void)applicationWillTerminate:(NSNotification *)aNotification {
  NSLog(@"%s", buffer);
}

@end
```

Thread 1: Use of deallocated memory



```
AddressSanitizerDemo PID 18763
CPU                                    0%
Memory                           Disabled
Energy Impact                        Zero
Disk                           Zero KB/s
Network                        Zero KB/s
Thread 1 Queue: com.apple.main-thread (serial)
    0 __asan::AsanDie()
    10 NSLog
    11 -[AppDelegate applicationWillTerminate:]
    27 main
    28 start
  0x6030000e8f90 1 byte inside a 32-byte heap regio...
    Memory deallocated by Thread 1
      0 wrap_free
      1 -[AppDelegate applicationDidFinishLaunching:]
      21 NSApplicationMain
      22 main
    Memory allocated by Thread 1
      0 wrap_malloc
      1 -[AppDelegate applicationDidFinishLaunching:]
      21 NSApplicationMain
      22 main
```

```
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

char *buffer;

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
  buffer = malloc(32);
  snprintf(buffer, 32, "Hello, World!");
  NSLog(@"%s", buffer);
  free(buffer);
}

- (void)applicationWillTerminate:(NSNotification *)aNotification {
  NSLog(@"%s", buffer);        Thread 1: Use of deallocated memory
}

@end
```



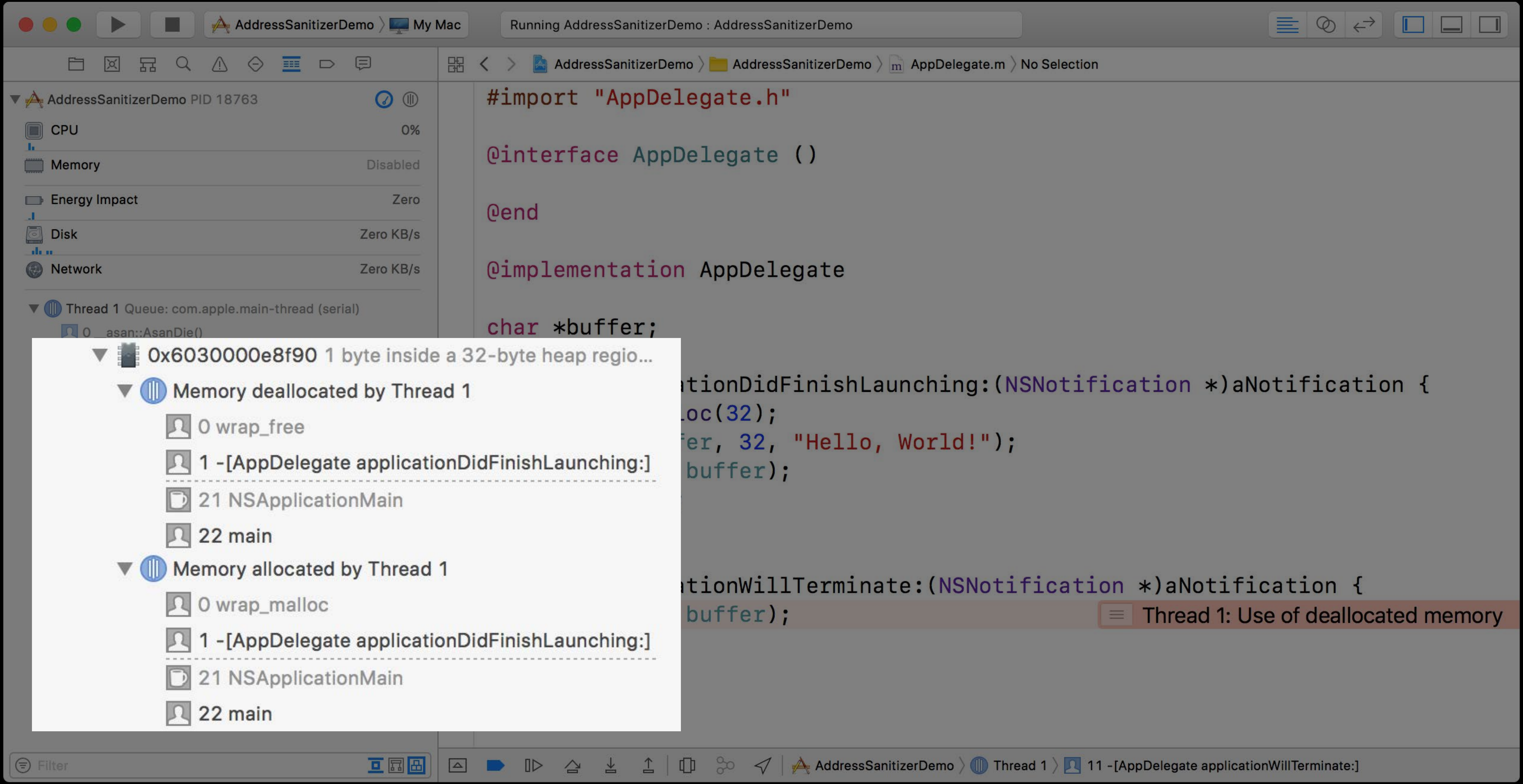
AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 AddressSanitizerDemo 〉 AppDelegate.m 〉 No Selection

AddressSanitizerDemo PID 18763
CPU 0%
Memory Disabled
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s

```
Thread 1 Queue: com.apple.main-thread (serial)
  0 __asan::AsanDie()
  10 NSLog
  11 -[AppDelegate applicationWillTerminate:]
  27 main
  28 start
0x6030000e8f90 1 byte inside a 32-byte heap regio...
  Memory deallocated by Thread 1
    0 wrap_free
    1 -[AppDelegate applicationDidFinishLaunching:]
    21 NSApplicationMain
    22 main
  Memory allocated by Thread 1
    0 wrap_malloc
    1 -[AppDelegate applicationDidFinishLaunching:]
    21 NSApplicationMain
    22 main
```

```
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

char *buffer;

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
  buffer = malloc(32);
  snprintf(buffer, 32, "Hello, World!");
  NSLog(@"%s", buffer);
  free(buffer);
}

- (void)applicationWillTerminate:(NSNotification *)aNotification {
  NSLog(@"%s", buffer);          Thread 1: Use of deallocated memory
}

@end
```

Filter
AddressSanitizerDemo 〉 Thread 1 〉 11 -[AppDelegate applicationWillTerminate:]



```
AddressSanitizerDemo PID 18763
CPU                0%
Memory             Disabled
Energy Impact      Zero
Disk               Zero KB/s
Network            Zero KB/s
Thread 1 Queue: com.apple.main-thread (serial)
    0 __asan::AsanDie()
```

```
0x6030000e8f90 1 byte inside a 32-byte heap regio...
    Memory deallocated by Thread 1
        0 wrap_free
        1 -[AppDelegate applicationDidFinishLaunching:]
        21 NSApplicationMain
        22 main
    Memory allocated by Thread 1
        0 wrap_malloc
        1 -[AppDelegate applicationDidFinishLaunching:]
        21 NSApplicationMain
        22 main
```

```
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

char *buffer;

tionDidFinishLaunching:(NSNotification *)aNotification {
oc(32);
fer, 32, "Hello, World!");
 buffer);

tionWillTerminate:(NSNotification *)aNotification {
 buffer);
```

Thread 1: Use of deallocated memory

Address Sanitizer in Xcode 9

NEW

Detects use-after-scope

Detects use-after-return (opt-in)

Compatible with Malloc Scribble

```
// Use of Stack Memory Out of Scope

int *integer_pointer = NULL;
if (is_some_condition_true()) {
  int value = calculate_value();
  integer_pointer = &value;
}
*integer_pointer = 42;
```

✕

❌

```
// Use of Stack Memory Out of Scope

int *integer_pointer = NULL;
if (is_some_condition_true()) {
  int value = calculate_value();
  integer_pointer = &value;
}
*integer_pointer = 42;
```

```
// Use of Stack Memory Out of Scope

int *integer_pointer = NULL;
if (is_some_condition_true()) {
  int value = calculate_value();
  integer_pointer = &value;
}
*integer_pointer = 42;
```

✕

```
// Use of Stack Memory Out of Scope

int *integer_pointer = NULL;
if (is_some_condition_true()) {
  int value = calculate_value();
  integer_pointer = &value;
}
*integer_pointer = 42;
```

```
// Use of Stack Memory Out of Scope

int *integer_pointer = NULL;
if (is_some_condition_true()) {
  int value = calculate_value();
  integer_pointer = &value;
}
*integer_pointer = 42;
```

Use of out of scope stack memory

```c
// Use of Stack Memory after Return

int *returns_address_of_stack() {
  int a = 42;
  return &a;
}

int *integer_pointer = returns_address_of_stack();
*integer_pointer = 43;
```

❌

```
// Use of Stack Memory after Return

int *returns_address_of_stack() {
  int a = 42;
  return &a;
}

int *integer_pointer = returns_address_of_stack();
*integer_pointer = 43;
```

```
// Use of Stack Memory after Return

int *returns_address_of_stack() {
  int a = 42;
  return &a;
}

int *integer_pointer = returns_address_of_stack();
*integer_pointer = 43;
```

```
// Use of Stack Memory after Return

int *returns_address_of_stack() {
  int a = 42;
  return &a;
}

int *integer_pointer = returns_address_of_stack();
*integer_pointer = 43;
```

```c
// Use of Stack Memory after Return

int *returns_address_of_stack() {
  int a = 42;
  return &a;
}

int *integer_pointer = returns_address_of_stack();
*integer_pointer = 43;
```

Use of stack memory after return

Address Sanitizer and Swift

Swift is a much safer language

Mixed projects

Unsafe pointer types are not memory safe

```
// Use-after-free Bug Using UnsafePointer

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

```
// Use-after-free Bug Using UnsafePointer

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

```swift
// Use-after-free Bug Using UnsafePointer

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

```swift
// Use-after-free Bug Using UnsafePointer

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

```
// Use-after-free Bug Using UnsafePointer

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

```
// Use-after-free Bug Using UnsafePointer

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

```
// Use-after-free Bug Using UnsafePointer

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

Use of deallocated memory

```swift
// Use UnsafePointer Only Inside the Closure

let string = "Hello, World!"
var firstBytePointer: UnsafePointer<CChar>
…
string.withCString { pointerToCString in
  firstBytePointer = pointerToCString
}
…
let firstByte = firstBytePointer.pointee
print(firstByte)
```

```swift
// Use UnsafePointer Only Inside the Closure

let string = "Hello, World!"

string.withCString { pointerToCString in
  var firstBytePointer: UnsafePointer<CChar>
  firstBytePointer = pointerToCString
  …
  let firstByte = firstBytePointer.pointee
  print(firstByte)
}
```

```swift
// Use UnsafePointer Only Inside the Closure

let string = "Hello, World!"

string.withCString { pointerToCString in
  var firstBytePointer: UnsafePointer<CChar>
  firstBytePointer = pointerToCString
  …
  let firstByte = firstBytePointer.pointee
  print(firstByte)
}
```

```swift
// Use UnsafePointer Only Inside the Closure

let string = "Hello, World!"

string.withCString { pointerToCString in
  var firstBytePointer: UnsafePointer<CChar>
  firstBytePointer = pointerToCString
  …
  let firstByte = firstBytePointer.pointee
  print(firstByte)
}
```

```swift
// Use UnsafePointer Only Inside the Closure

let string = "Hello, World!"

string.withCString { pointerToCString in
  …
  let firstByte = pointerToCString.pointee
  print(firstByte)
}
```

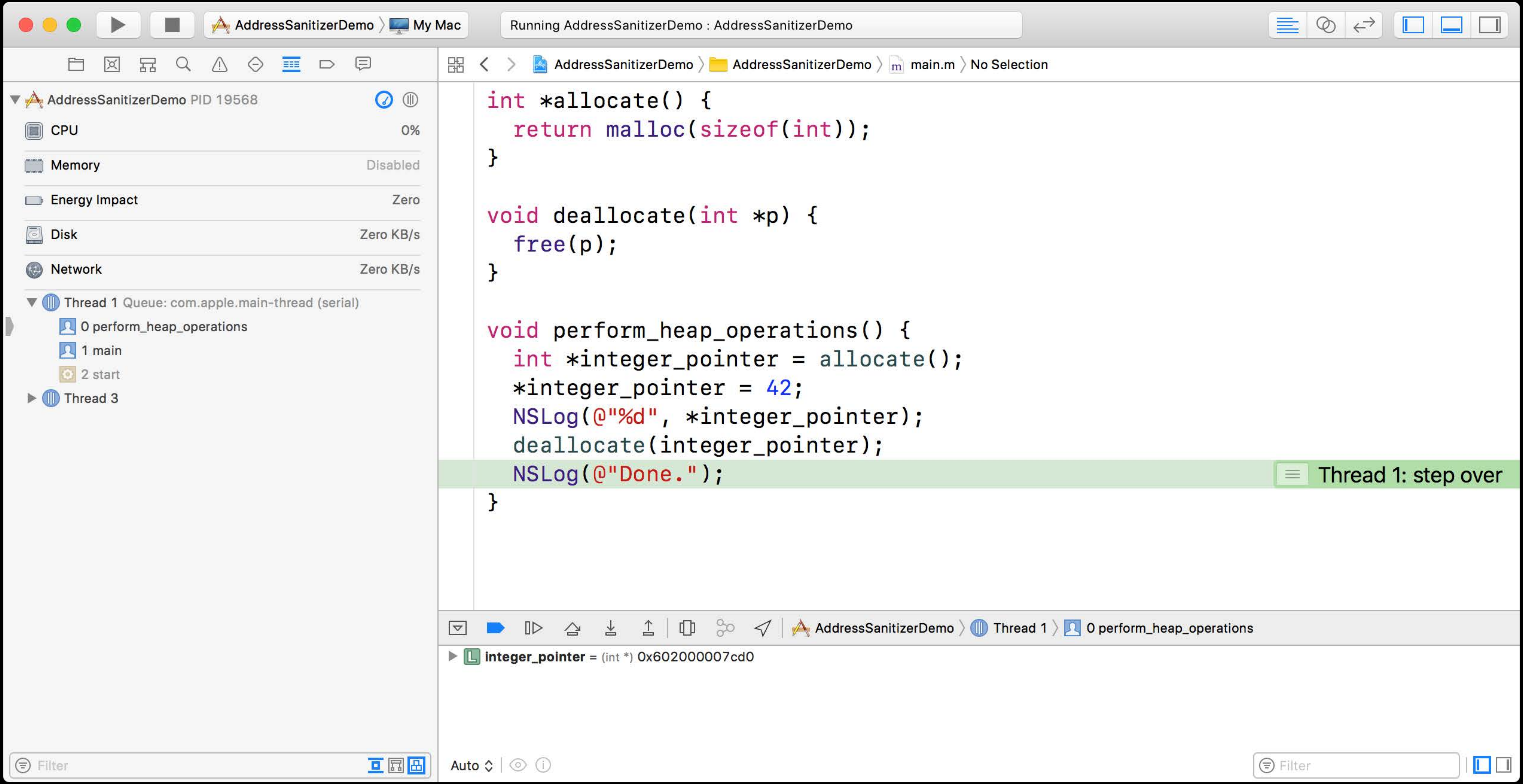
✓

Better Debugging Experience

Makes debugging easier

Allocation and deallocation backtraces

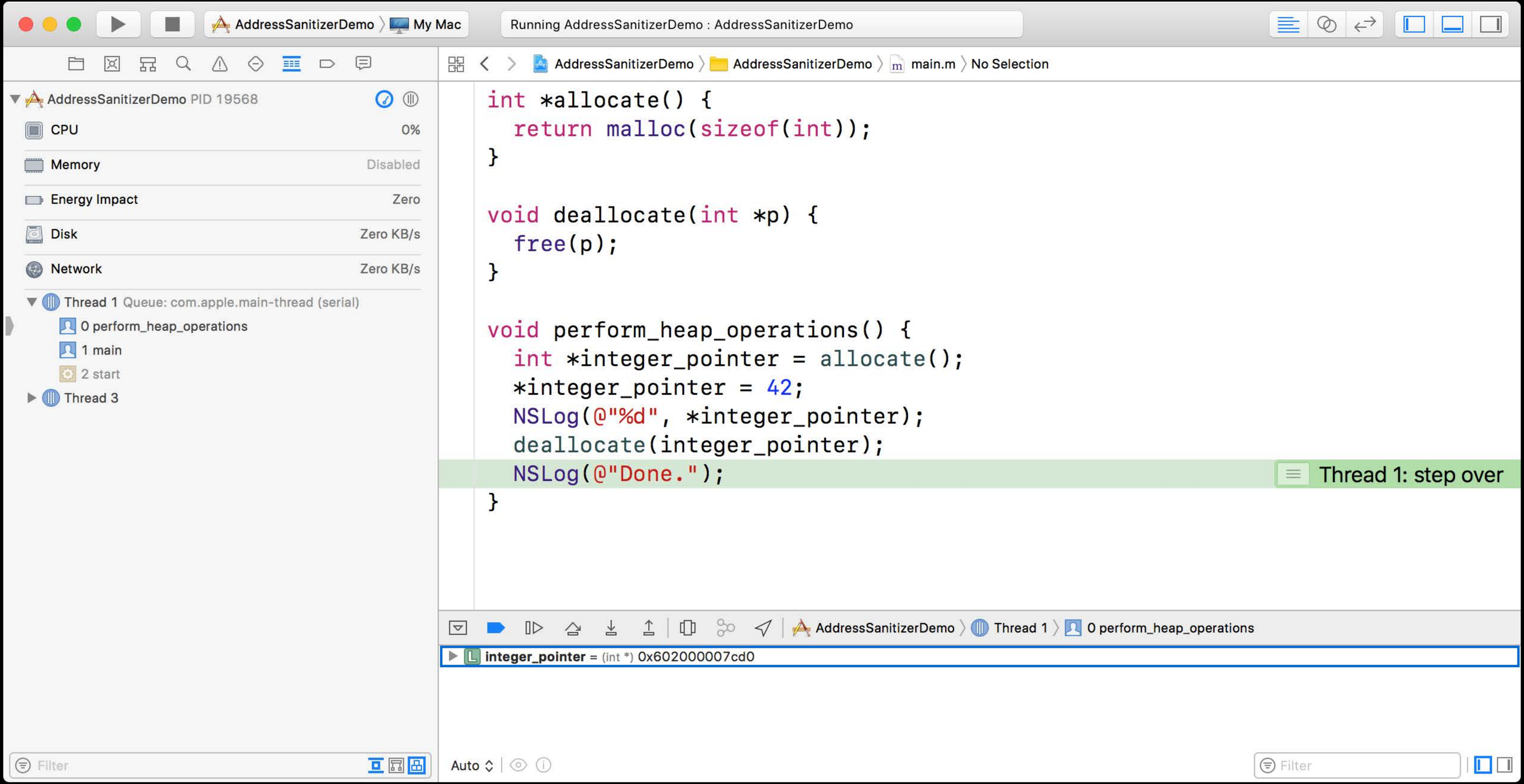
Shows valid and invalid bytes of memory



AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 AddressSanitizerDemo 〉 main.m 〉 No Selection
AddressSanitizerDemo PID 19568
CPU 0%
Memory Disabled
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: com.apple.main-thread (serial)
0 perform_heap_operations
1 main
2 start
Thread 3
Thread 1: step over
AddressSanitizerDemo 〉 Thread 1 〉 0 perform_heap_operations
Auto
Filter
Filter

```
int *allocate() {
  return malloc(sizeof(int));
}

void deallocate(int *p) {
  free(p);
}

void perform_heap_operations() {
  int *integer_pointer = allocate();
  *integer_pointer = 42;
  NSLog(@"%d", *integer_pointer);
  deallocate(integer_pointer);
  NSLog(@"Done.");
}
```

integer_pointer = (int *) 0x602000007cd0


AddressSanitizerDemo
My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 AddressSanitizerDemo 〉 main.m 〉 No Selection
Thread 1: step over
AddressSanitizerDemo 〉 Thread 1 〉 0 perform_heap_operations
Filter
Auto
Filter

AddressSanitizerDemo PID 19568

CPU 0%

Memory Disabled

Energy Impact Zero

Disk Zero KB/s

Network Zero KB/s

Thread 1 Queue: com.apple.main-thread (serial)

0 perform_heap_operations

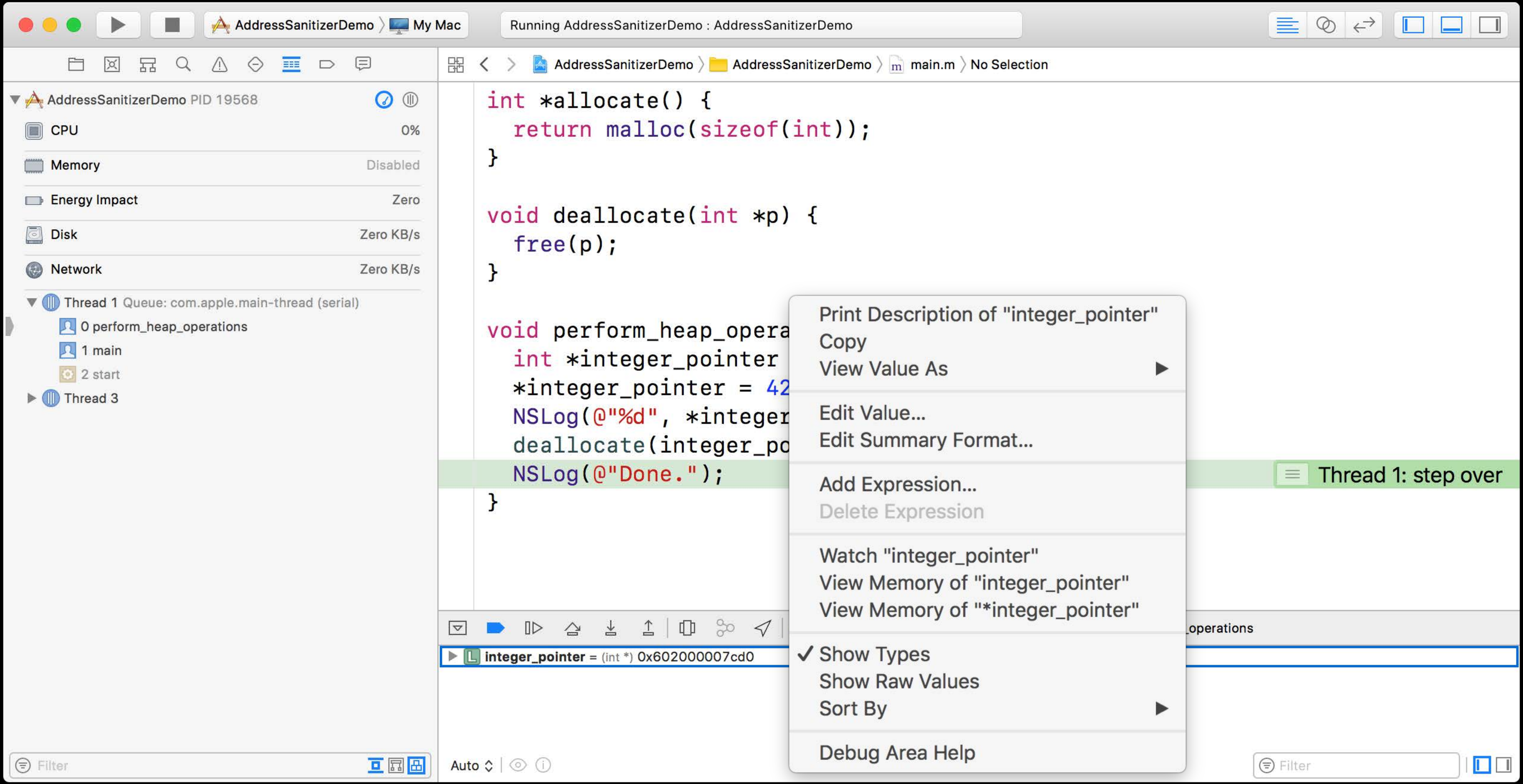
1 main

2 start

Thread 3

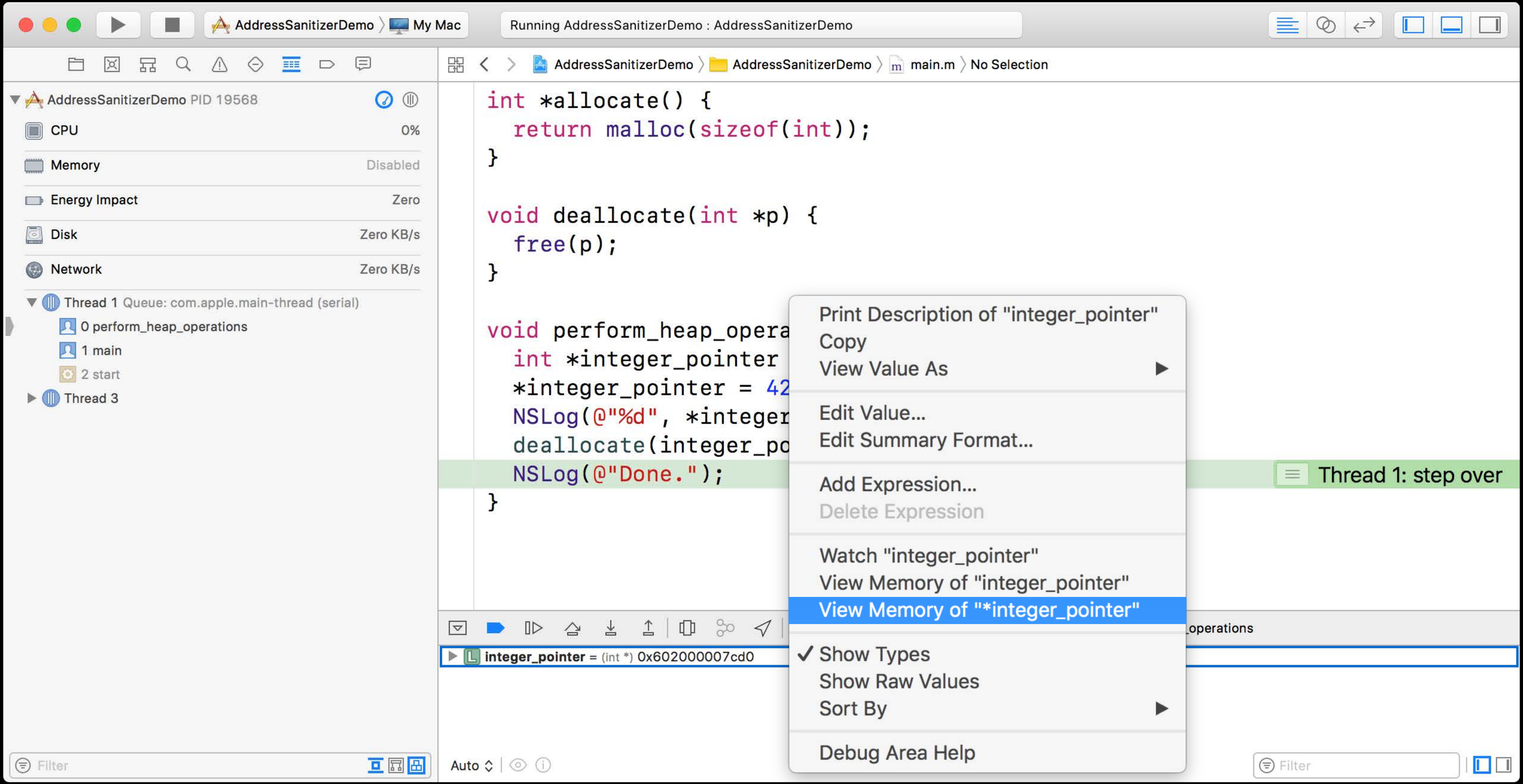
```
int *allocate() {
  return malloc(sizeof(int));
}

void deallocate(int *p) {
  free(p);
}

void perform_heap_operations() {
  int *integer_pointer = allocate();
  *integer_pointer = 42;
  NSLog(@"%d", *integer_pointer);
  deallocate(integer_pointer);
  NSLog(@"Done.");
}
```

```
integer_pointer = (int *) 0x602000007cd0
```



```
int *allocate() {
  return malloc(sizeof(int));
}

void deallocate(int *p) {
  free(p);
}

void perform_heap_opera
  int *integer_pointer
  *integer_pointer = 42
  NSLog(@"%d", *integer
  deallocate(integer_po
  NSLog(@"Done.");
}
```

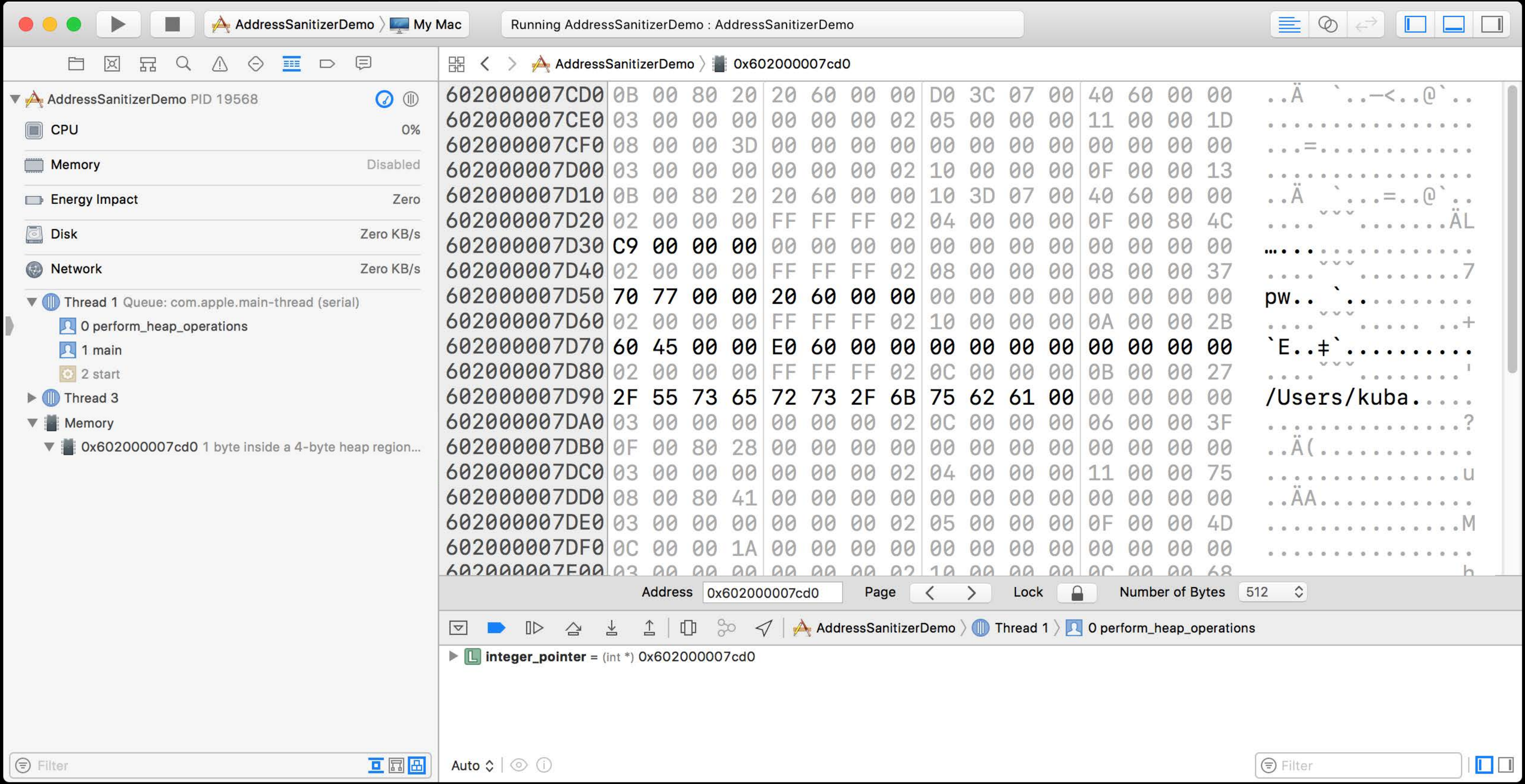
```
integer_pointer = (int *) 0x602000007cd0
```



AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 AddressSanitizerDemo 〉 main.m 〉 No Selection
AddressSanitizerDemo PID 19568
CPU 0%
Memory Disabled
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: com.apple.main-thread (serial)
0 perform_heap_operations
1 main
2 start
Thread 3
Print Description of "integer_pointer"
Copy
View Value As ▶
Edit Value…
Edit Summary Format…
Add Expression…
Delete Expression
Watch "integer_pointer"
View Memory of "integer_pointer"
View Memory of "*integer_pointer"
✓ Show Types
Show Raw Values
Sort By ▶
Debug Area Help
Thread 1: step over
_operations
Auto
Filter
Filter

```
int *allocate() {
  return malloc(sizeof(int));
}

void deallocate(int *p) {
  free(p);
}

void perform_heap_opera
  int *integer_pointer
  *integer_pointer = 42
  NSLog(@"%d", *integer
  deallocate(integer_po
  NSLog(@"Done.");
}
```

```
integer_pointer = (int *) 0x602000007cd0
```


AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 0x602000007cd0

AddressSanitizerDemo PID 19568

CPU 0%

Memory Disabled

Energy Impact Zero

Disk Zero KB/s

Network Zero KB/s

Thread 1 Queue: com.apple.main-thread (serial)

0 perform_heap_operations

1 main

2 start

Thread 3

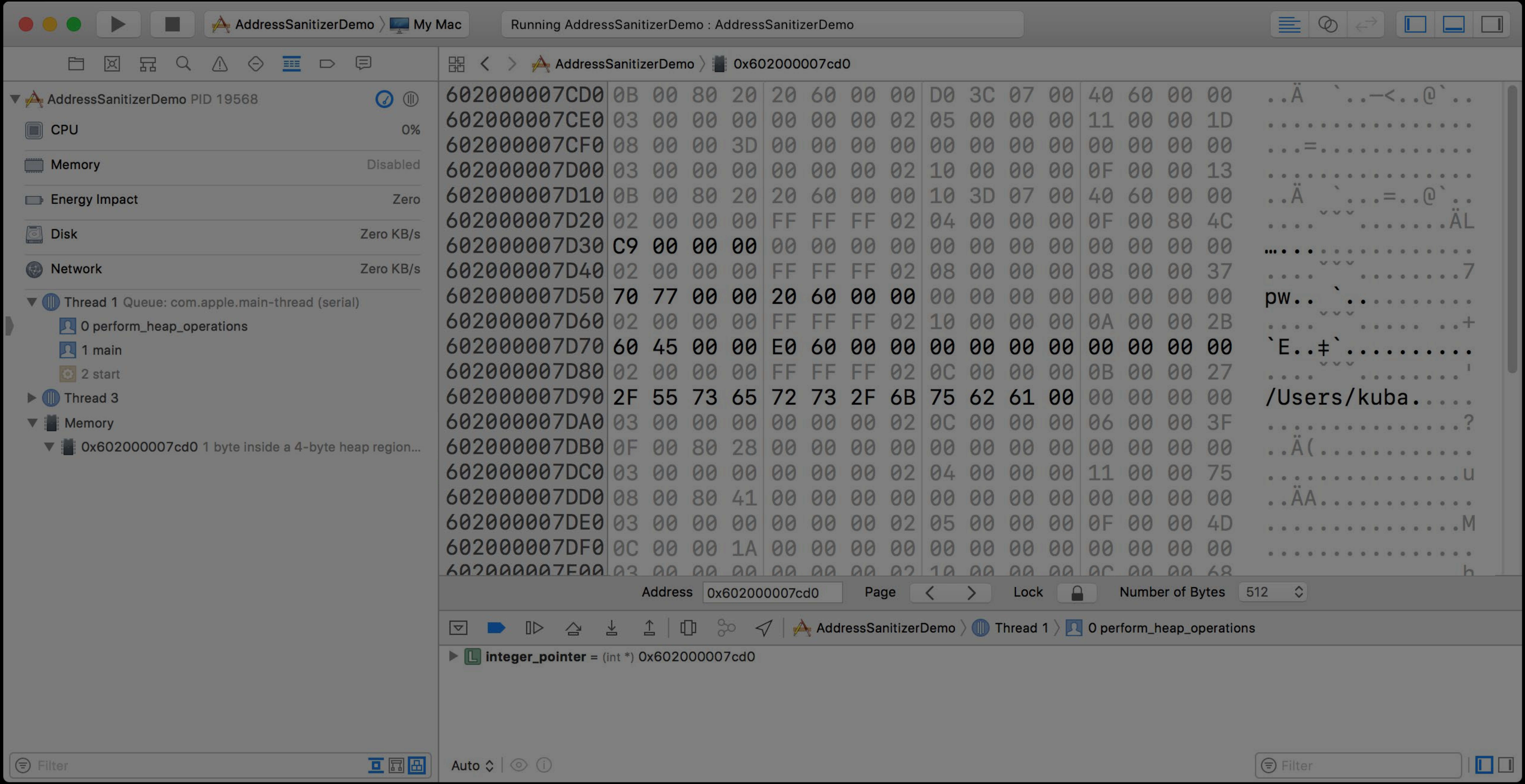
Memory

0x602000007cd0 1 byte inside a 4-byte heap region...

```
602000007CD0 0B 00 80 20 20 60 00 00 D0 3C 07 00 40 60 00 00  ..Ä  `..—<..@`..
602000007CE0 03 00 00 00 00 00 00 02 05 00 00 00 11 00 00 1D  ................
602000007CF0 08 00 00 3D 00 00 00 00 00 00 00 00 00 00 00 00  ...=............
602000007D00 03 00 00 00 00 00 00 02 10 00 00 00 0F 00 00 13  ................
602000007D10 0B 00 80 20 20 60 00 00 10 3D 07 00 40 60 00 00  ..Ä  `...=..@`..
602000007D20 02 00 00 00 FF FF FF 02 04 00 00 00 0F 00 80 4C  ....˘˘˘.......ÄL
602000007D30 C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  …...............
602000007D40 02 00 00 00 FF FF FF 02 08 00 00 00 08 00 00 37  ....˘˘˘........7
602000007D50 70 77 00 00 20 60 00 00 00 00 00 00 00 00 00 00  pw.. `..........
602000007D60 02 00 00 00 FF FF FF 02 10 00 00 00 0A 00 00 2B  ....˘˘˘..... ..+
602000007D70 60 45 00 00 E0 60 00 00 00 00 00 00 00 00 00 00  `E..‡`..........
602000007D80 02 00 00 00 FF FF FF 02 0C 00 00 00 0B 00 00 27  ....˘˘˘........'
602000007D90 2F 55 73 65 72 73 2F 6B 75 62 61 00 00 00 00 00  /Users/kuba.....
602000007DA0 03 00 00 00 00 00 00 02 0C 00 00 00 06 00 00 3F  ...............?
602000007DB0 0F 00 80 28 00 00 00 00 00 00 00 00 00 00 00 00  ..Ä(............
602000007DC0 03 00 00 00 00 00 00 02 04 00 00 00 11 00 00 75  ...............u
602000007DD0 08 00 80 41 00 00 00 00 00 00 00 00 00 00 00 00  ..ÄA............
602000007DE0 03 00 00 00 00 00 00 02 05 00 00 00 0F 00 00 4D  ...............M
602000007DF0 0C 00 00 1A 00 00 00 00 00 00 00 00 00 00 00 00  ................
602000007E00 03 00 00 00 00 00 00 02 10 00 00 00 0C 00 00 68                 h
```

Address 0x602000007cd0 Page Lock Number of Bytes 512

integer_pointer = (int *) 0x602000007cd0



AddressSanitizerDemo PID 19568

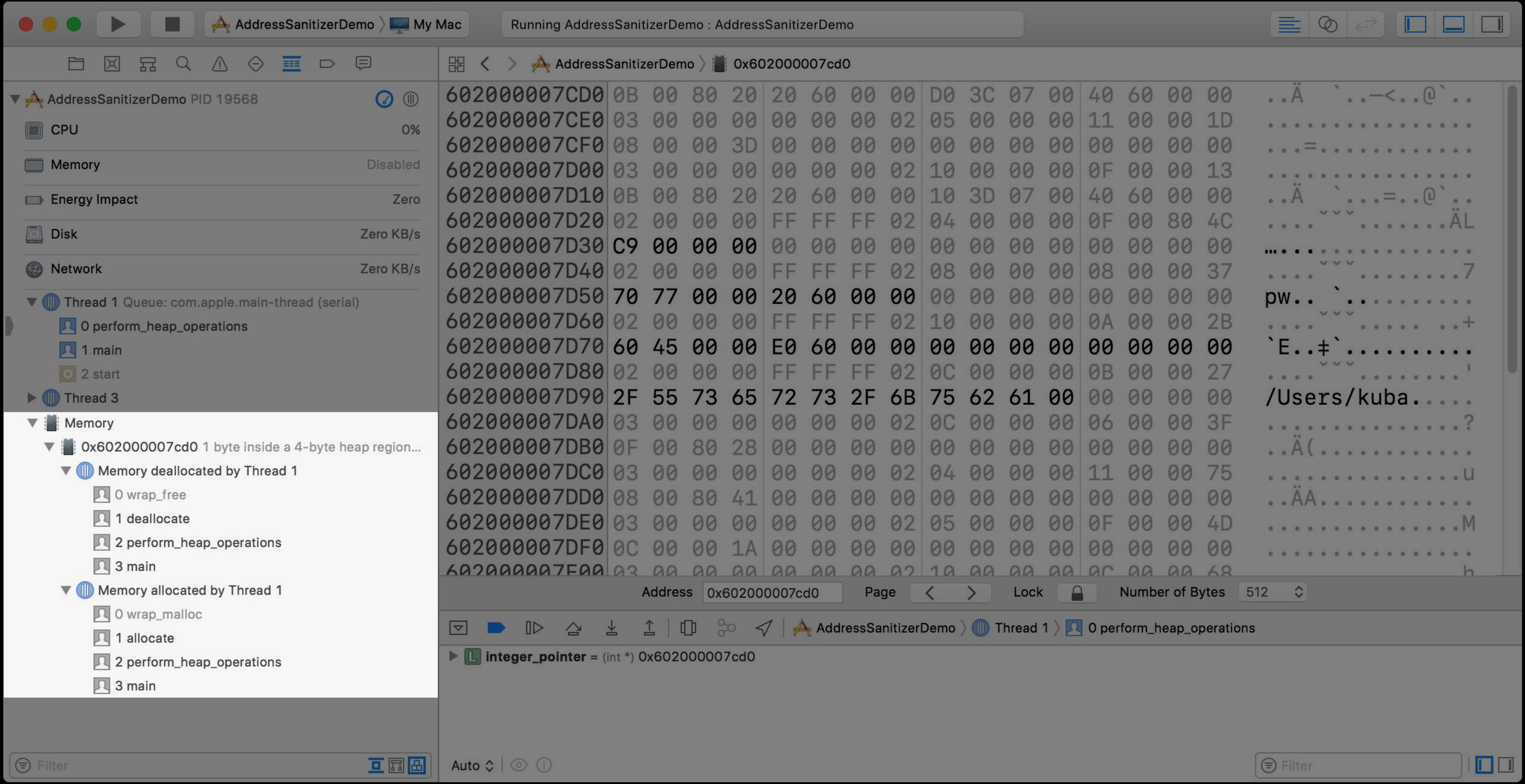
CPU	0%
Memory	Disabled
Energy Impact	Zero
Disk	Zero KB/s
Network	Zero KB/s

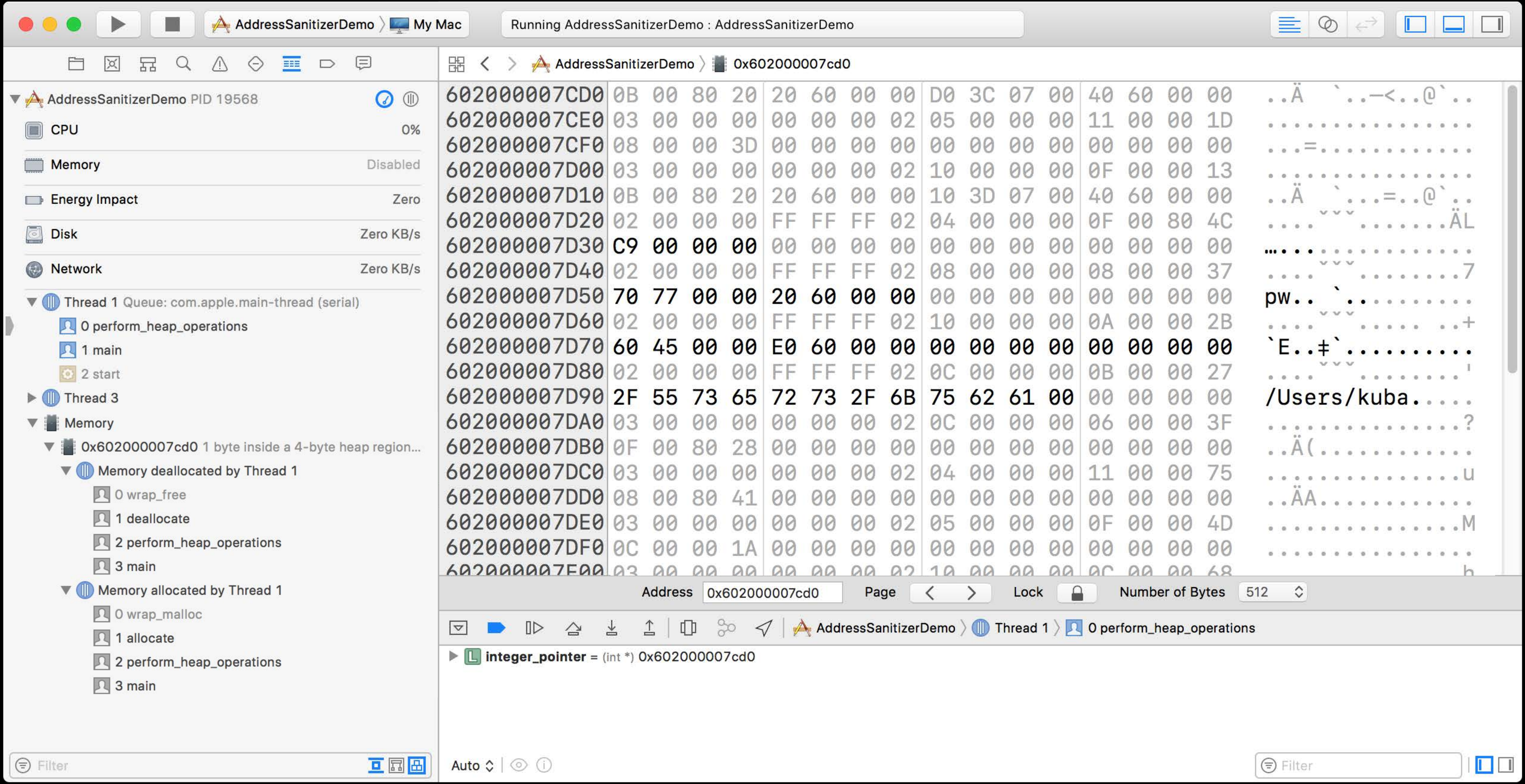
Thread 1 Queue: com.apple.main-thread (serial)
0 perform_heap_operations
1 main
2 start
Thread 3
Memory
0x602000007cd0 1 byte inside a 4-byte heap region...

```
602000007CD0 0B 00 80 20 20 60 00 00 D0 3C 07 00 40 60 00 00  ..Ä  `..—<..@`..
602000007CE0 03 00 00 00 00 00 00 02 05 00 00 00 11 00 00 1D  ................
602000007CF0 08 00 00 3D 00 00 00 00 00 00 00 00 00 00 00 00  ...=............
602000007D00 03 00 00 00 00 00 00 02 10 00 00 00 0F 00 00 13  ................
602000007D10 0B 00 80 20 20 60 00 00 10 3D 07 00 40 60 00 00  ..Ä  `...=..@`..
602000007D20 02 00 00 00 FF FF FF 02 04 00 00 00 0F 00 80 4C  ....˘˘˘.......ÄL
602000007D30 C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  …...............
602000007D40 02 00 00 00 FF FF FF 02 08 00 00 00 08 00 00 37  ....˘˘˘........7
602000007D50 70 77 00 00 20 60 00 00 00 00 00 00 00 00 00 00  pw.. `..........
602000007D60 02 00 00 00 FF FF FF 02 10 00 00 00 0A 00 00 2B  ....˘˘˘..... ..+
602000007D70 60 45 00 00 E0 60 00 00 00 00 00 00 00 00 00 00  `E..‡`..........
602000007D80 02 00 00 00 FF FF FF 02 0C 00 00 00 0B 00 00 27  ....˘˘˘........'
602000007D90 2F 55 73 65 72 73 2F 6B 75 62 61 00 00 00 00 00  /Users/kuba.....
602000007DA0 03 00 00 00 00 00 00 02 0C 00 00 00 06 00 00 3F  ...............?
602000007DB0 0F 00 80 28 00 00 00 00 00 00 00 00 00 00 00 00  ..Ä(............
602000007DC0 03 00 00 00 00 00 00 02 04 00 00 00 11 00 00 75  ...............u
602000007DD0 08 00 80 41 00 00 00 00 00 00 00 00 00 00 00 00  ..ÄA............
602000007DE0 03 00 00 00 00 00 00 02 05 00 00 00 0F 00 00 4D  ...............M
602000007DF0 0C 00 00 1A 00 00 00 00 00 00 00 00 00 00 00 00  ................
602000007E00 03 00 00 00 00 00 00 02 10 00 00 00 0C 00 00 68                 h
```

Address 0x602000007cd0 Page Lock Number of Bytes 512

integer_pointer = (int *) 0x602000007cd0





AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo

AddressSanitizerDemo PID 19568

CPU	0%
Memory	Disabled
Energy Impact	Zero
Disk	Zero KB/s
Network	Zero KB/s

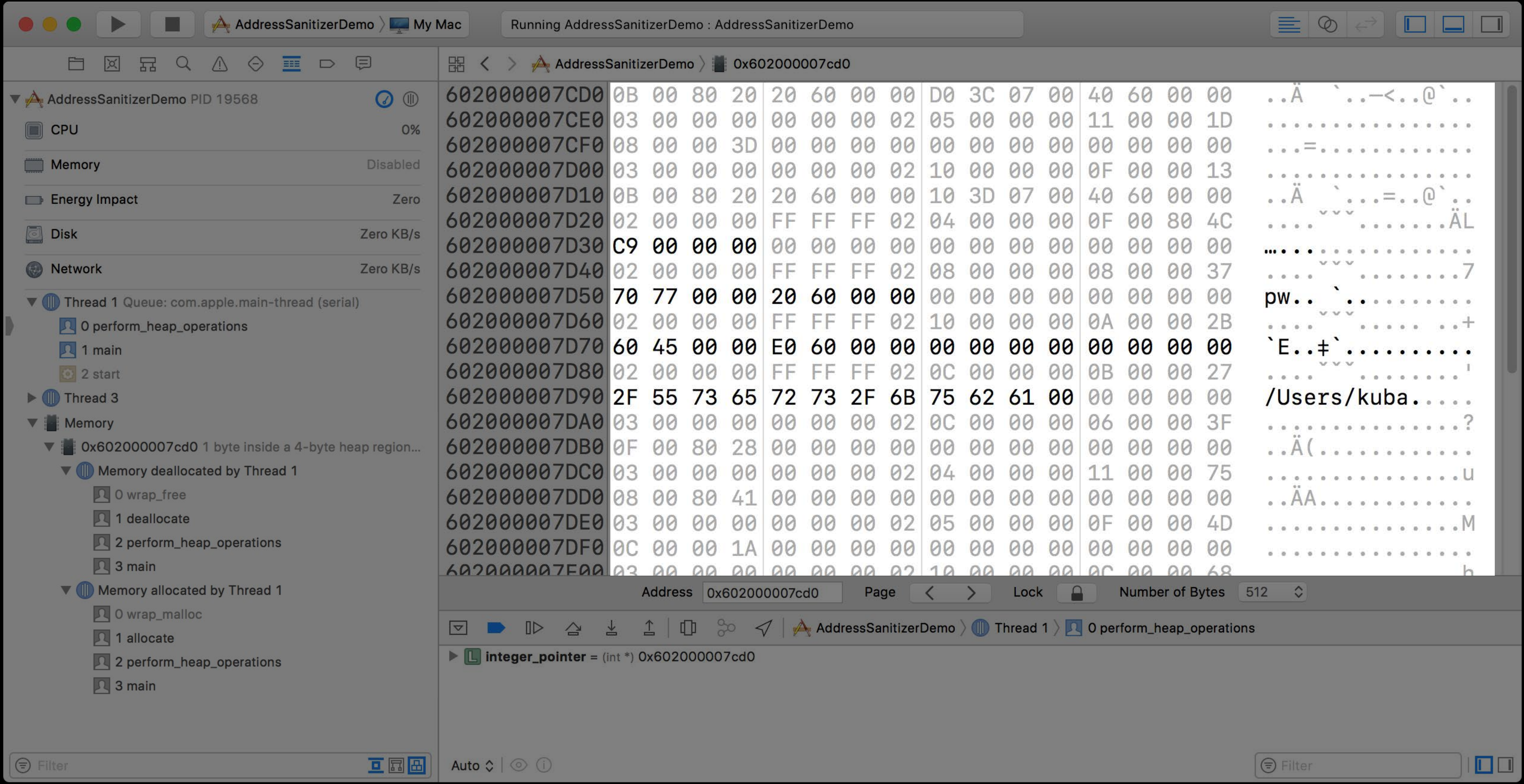
- Thread 1 Queue: com.apple.main-thread (serial)
  - 0 perform_heap_operations
  - 1 main
  - 2 start
- Thread 3
- Memory
  - 0x602000007cd0 1 byte inside a 4-byte heap region...
    - Memory deallocated by Thread 1
      - 0 wrap_free
      - 1 deallocate
      - 2 perform_heap_operations
      - 3 main
    - Memory allocated by Thread 1
      - 0 wrap_malloc
      - 1 allocate
      - 2 perform_heap_operations
      - 3 main

AddressSanitizerDemo 〉 0x602000007cd0

```
602000007CD0 0B 00 80 20 20 60 00 00 D0 3C 07 00 40 60 00 00  ..Ä  `..—<..@`..
602000007CE0 03 00 00 00 00 00 00 02 05 00 00 00 11 00 00 1D  ................
602000007CF0 08 00 00 3D 00 00 00 00 00 00 00 00 00 00 00 00  ...=............
602000007D00 03 00 00 00 00 00 00 02 10 00 00 00 0F 00 00 13  ................
602000007D10 0B 00 80 20 20 60 00 00 10 3D 07 00 40 60 00 00  ..Ä  `...=..@`..
602000007D20 02 00 00 00 FF FF FF 02 04 00 00 00 0F 00 80 4C  ....ˇˇˇ.......ÄL
602000007D30 C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  …...............
602000007D40 02 00 00 00 FF FF FF 02 08 00 00 00 08 00 00 37  ....ˇˇˇ........7
602000007D50 70 77 00 00 20 60 00 00 00 00 00 00 00 00 00 00  pw.. `..........
602000007D60 02 00 00 00 FF FF FF 02 10 00 00 00 0A 00 00 2B  ....ˇˇˇ..... ..+
602000007D70 60 45 00 00 E0 60 00 00 00 00 00 00 00 00 00 00  `E..‡`..........
602000007D80 02 00 00 00 FF FF FF 02 0C 00 00 00 0B 00 00 27  ....ˇˇˇ........'
602000007D90 2F 55 73 65 72 73 2F 6B 75 62 61 00 00 00 00 00  /Users/kuba.....
602000007DA0 03 00 00 00 00 00 00 02 0C 00 00 00 06 00 00 3F  ...............?
602000007DB0 0F 00 80 28 00 00 00 00 00 00 00 00 00 00 00 00  ..Ä(............
602000007DC0 03 00 00 00 00 00 00 02 04 00 00 00 11 00 00 75  ...............u
602000007DD0 08 00 80 41 00 00 00 00 00 00 00 00 00 00 00 00  ..ÄA............
602000007DE0 03 00 00 00 00 00 00 02 05 00 00 00 0F 00 00 4D  ...............M
602000007DF0 0C 00 00 1A 00 00 00 00 00 00 00 00 00 00 00 00  ................
602000007E00 03 00 00 00 00 00 00 02 10 00 00 00 0C 00 00 68  ...............h
```

Address: 0x602000007cd0 Page Lock Number of Bytes: 512

integer_pointer = (int *) 0x602000007cd0



AddressSanitizerDemo 〉 My Mac
Running AddressSanitizerDemo : AddressSanitizerDemo
AddressSanitizerDemo 〉 0x602000007cd0

AddressSanitizerDemo PID 19568

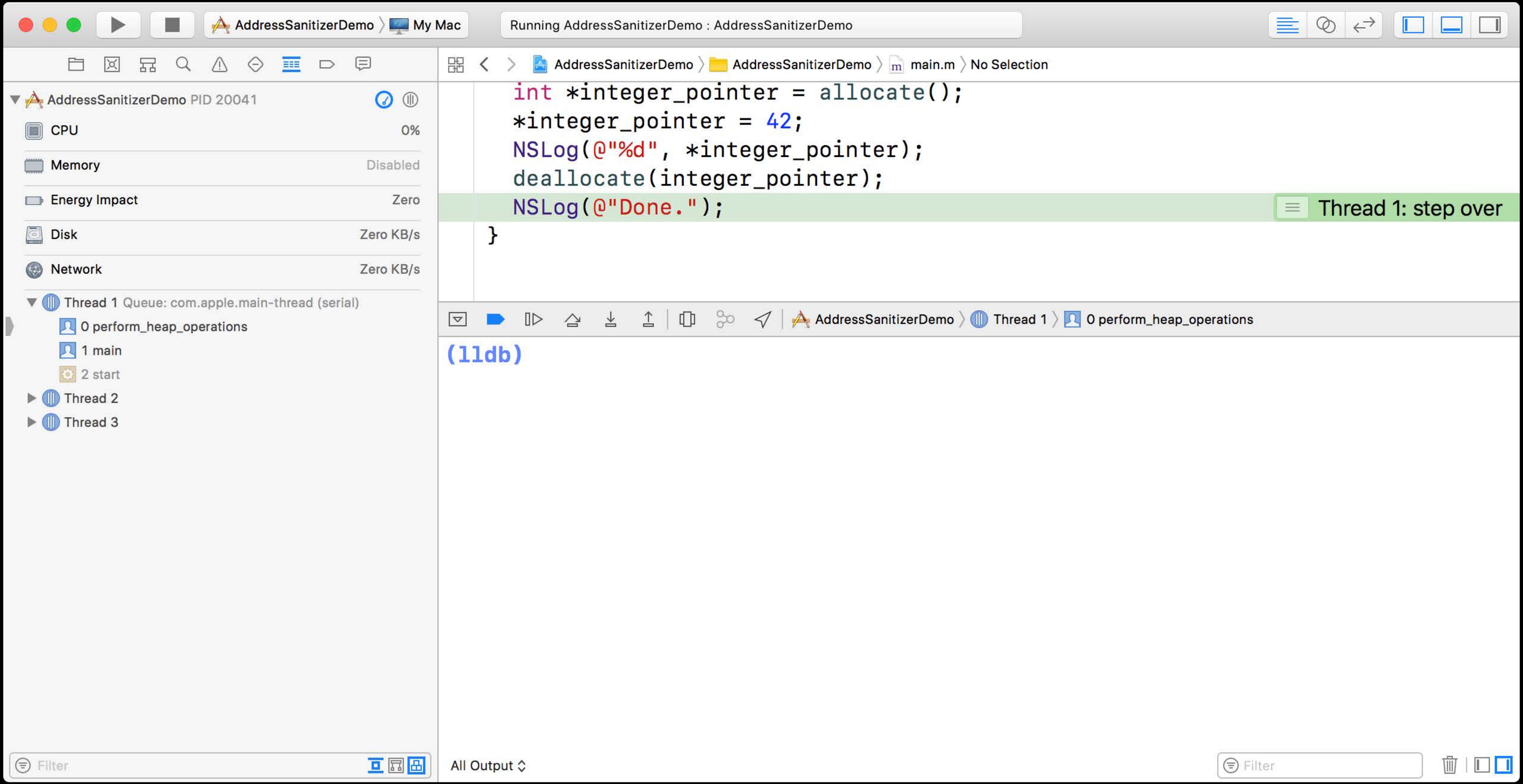
CPU	0%
Memory	Disabled
Energy Impact	Zero
Disk	Zero KB/s
Network	Zero KB/s

- Thread 1 Queue: com.apple.main-thread (serial)
  - 0 perform_heap_operations
  - 1 main
  - 2 start
- Thread 3
- Memory
  - 0x602000007cd0 1 byte inside a 4-byte heap region...
    - Memory deallocated by Thread 1
      - 0 wrap_free
      - 1 deallocate
      - 2 perform_heap_operations
      - 3 main
    - Memory allocated by Thread 1
      - 0 wrap_malloc
      - 1 allocate
      - 2 perform_heap_operations
      - 3 main

```
602000007CD0 0B 00 80 20 20 60 00 00 D0 3C 07 00 40 60 00 00   ..Ä  `..—<..@`..
602000007CE0 03 00 00 00 00 00 00 02 05 00 00 00 11 00 00 1D   ................
602000007CF0 08 00 00 3D 00 00 00 00 00 00 00 00 00 00 00 00   ...=............
602000007D00 03 00 00 00 00 00 00 02 10 00 00 00 0F 00 00 13   ................
602000007D10 0B 00 80 20 20 60 00 00 10 3D 07 00 40 60 00 00   ..Ä  `...=..@`..
602000007D20 02 00 00 00 FF FF FF 02 04 00 00 00 0F 00 80 4C   ....˘˘˘.......ÄL
602000007D30 C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   …...............
602000007D40 02 00 00 00 FF FF FF 02 08 00 00 00 08 00 00 37   ....˘˘˘........7
602000007D50 70 77 00 00 20 60 00 00 00 00 00 00 00 00 00 00   pw.. `..........
602000007D60 02 00 00 00 FF FF FF 02 10 00 00 00 0A 00 00 2B   ....˘˘˘..... ..+
602000007D70 60 45 00 00 E0 60 00 00 00 00 00 00 00 00 00 00   `E..‡`..........
602000007D80 02 00 00 00 FF FF FF 02 0C 00 00 00 0B 00 00 27   ....˘˘˘........'
602000007D90 2F 55 73 65 72 73 2F 6B 75 62 61 00 00 00 00 00   /Users/kuba.....
602000007DA0 03 00 00 00 00 00 00 02 0C 00 00 00 06 00 00 3F   ...............?
602000007DB0 0F 00 80 28 00 00 00 00 00 00 00 00 00 00 00 00   ..Ä(............
602000007DC0 03 00 00 00 00 00 00 02 04 00 00 00 11 00 00 75   ...............u
602000007DD0 08 00 80 41 00 00 00 00 00 00 00 00 00 00 00 00   ..ÄA............
602000007DE0 03 00 00 00 00 00 00 02 05 00 00 00 0F 00 00 4D   ...............M
602000007DF0 0C 00 00 1A 00 00 00 00 00 00 00 00 00 00 00 00   ................
602000007E00 03 00 00 00 00 00 00 02 10 00 00 00 0C 00 00 68   ...............h
```

Address 0x602000007cd0 Page Lock Number of Bytes 512

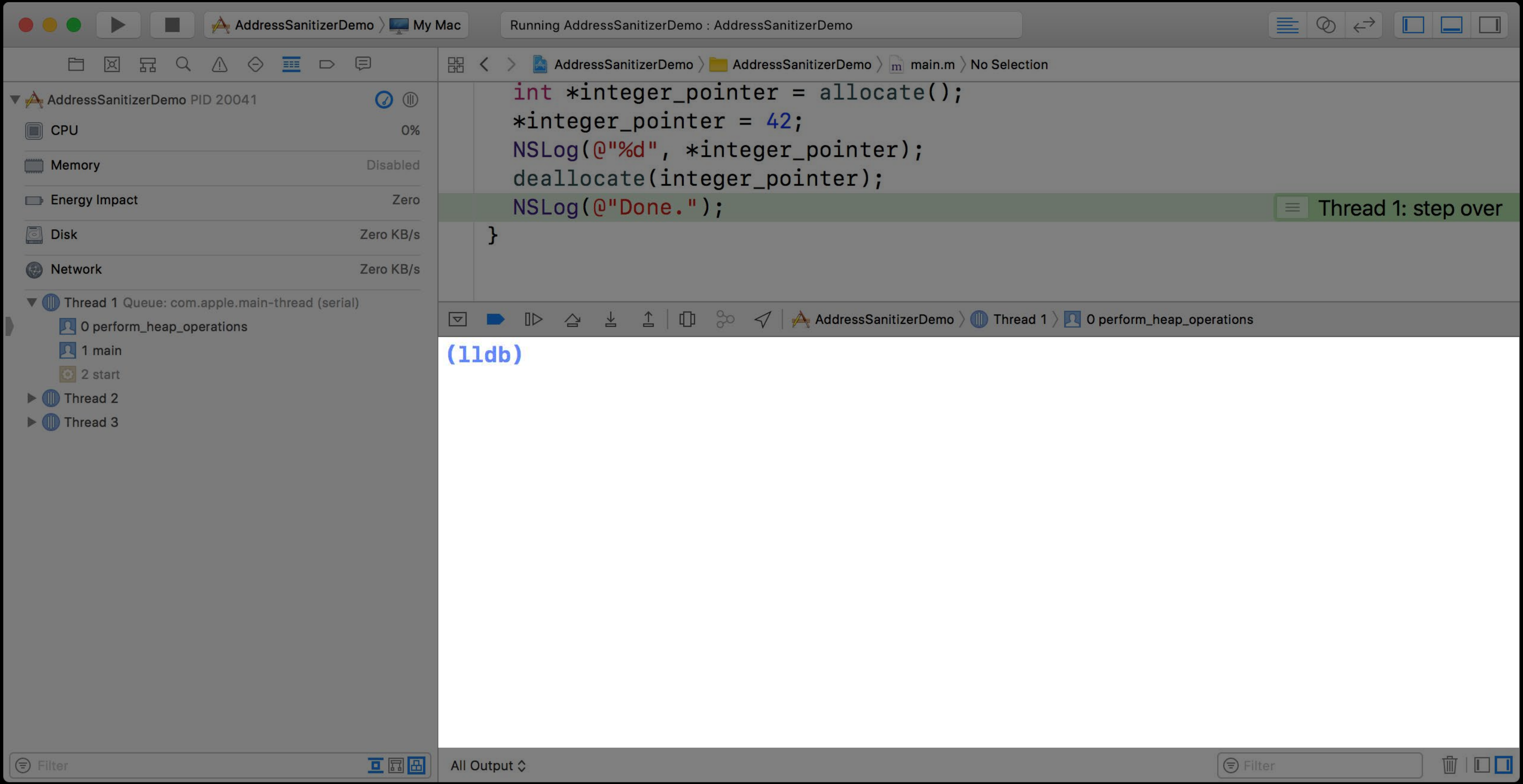
integer_pointer = (int *) 0x602000007cd0



```
    int *integer_pointer = allocate();
    *integer_pointer = 42;
    NSLog(@"%d", *integer_pointer);
    deallocate(integer_pointer);
    NSLog(@"Done.");
}
```

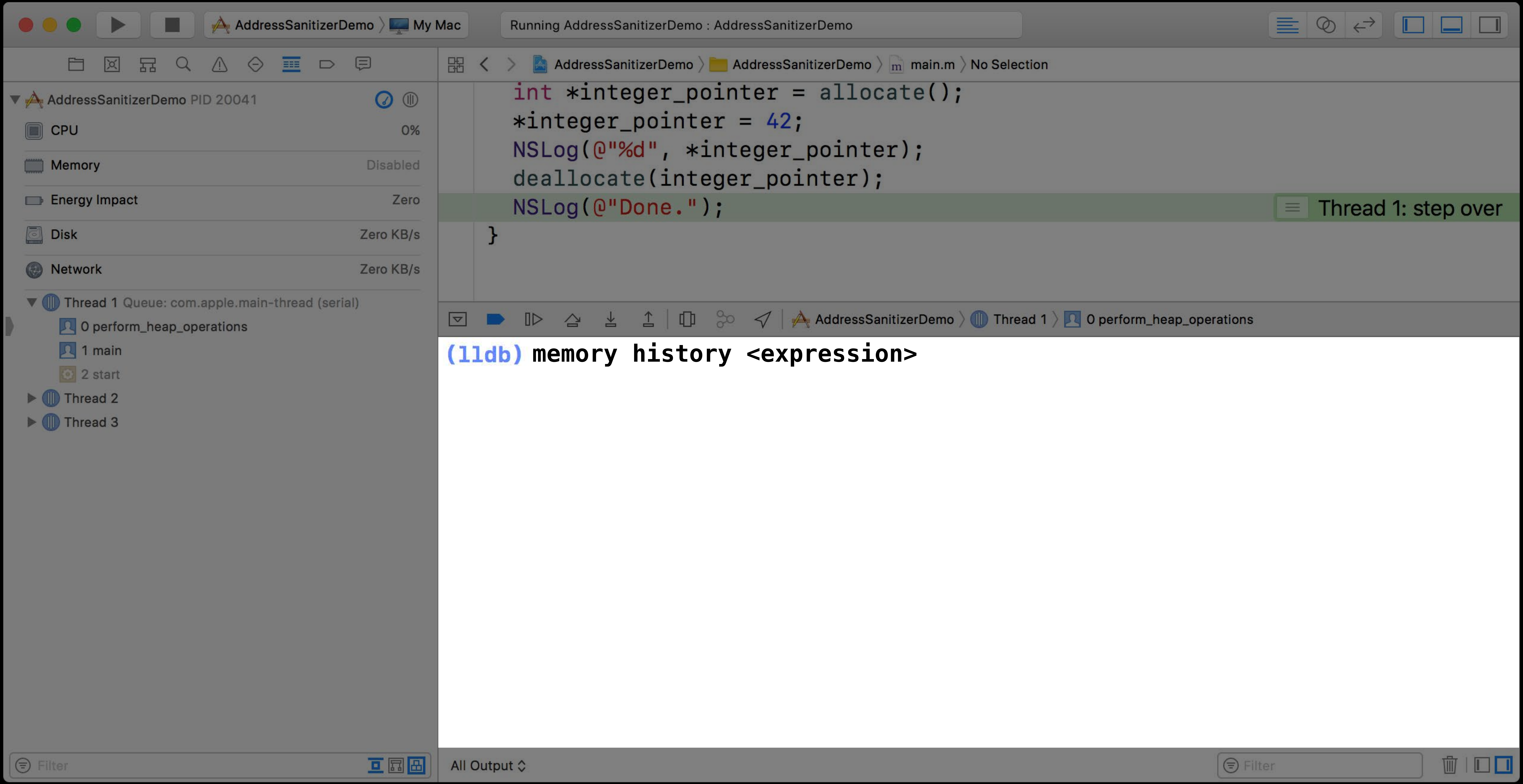
Thread 1: step over

```
(lldb)
```



```
    int *integer_pointer = allocate();
    *integer_pointer = 42;
    NSLog(@"%d", *integer_pointer);
    deallocate(integer_pointer);
    NSLog(@"Done.");
}
```

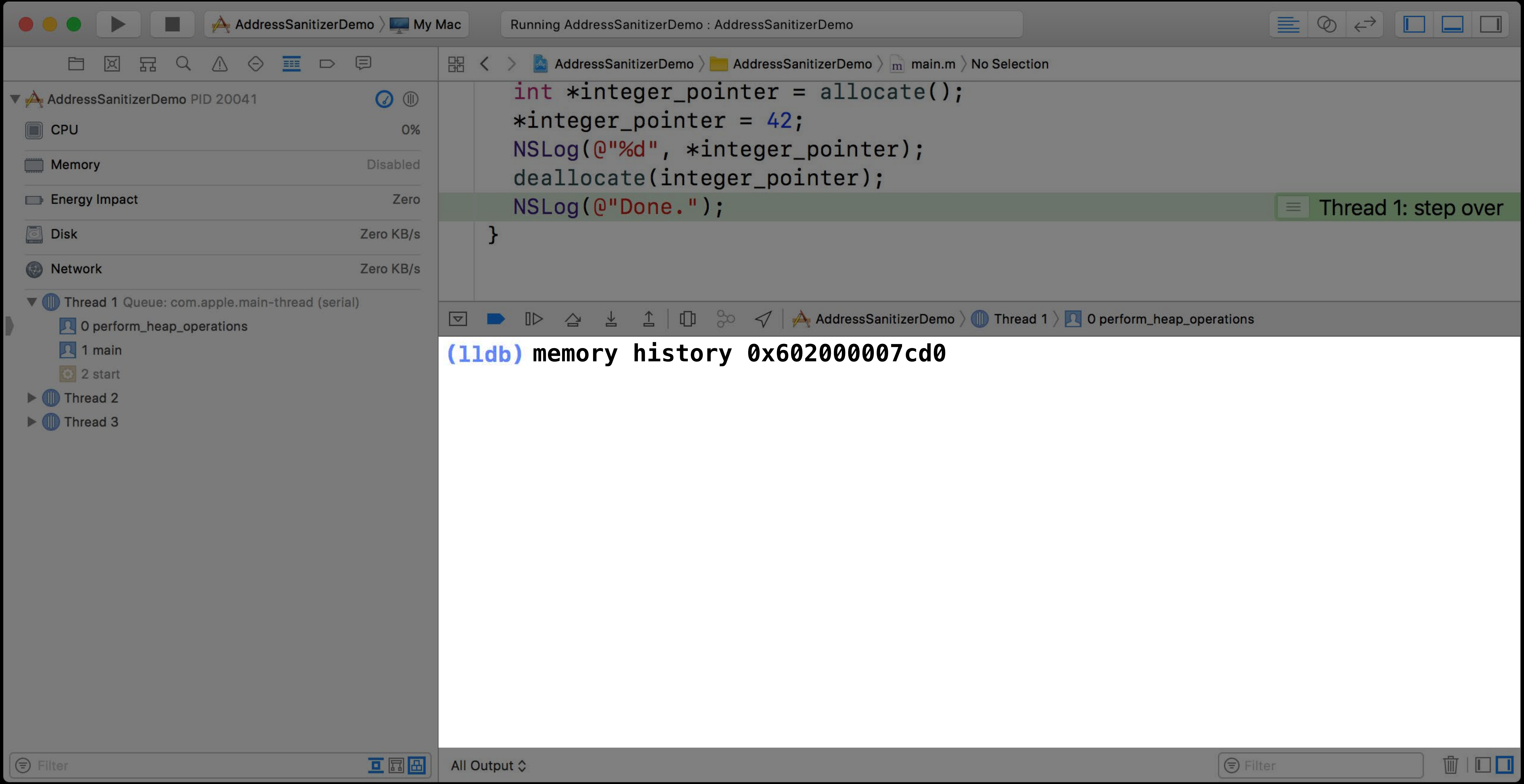
```
(lldb)
```



```
    int *integer_pointer = allocate();
    *integer_pointer = 42;
    NSLog(@"%d", *integer_pointer);
    deallocate(integer_pointer);
    NSLog(@"Done.");
}
```

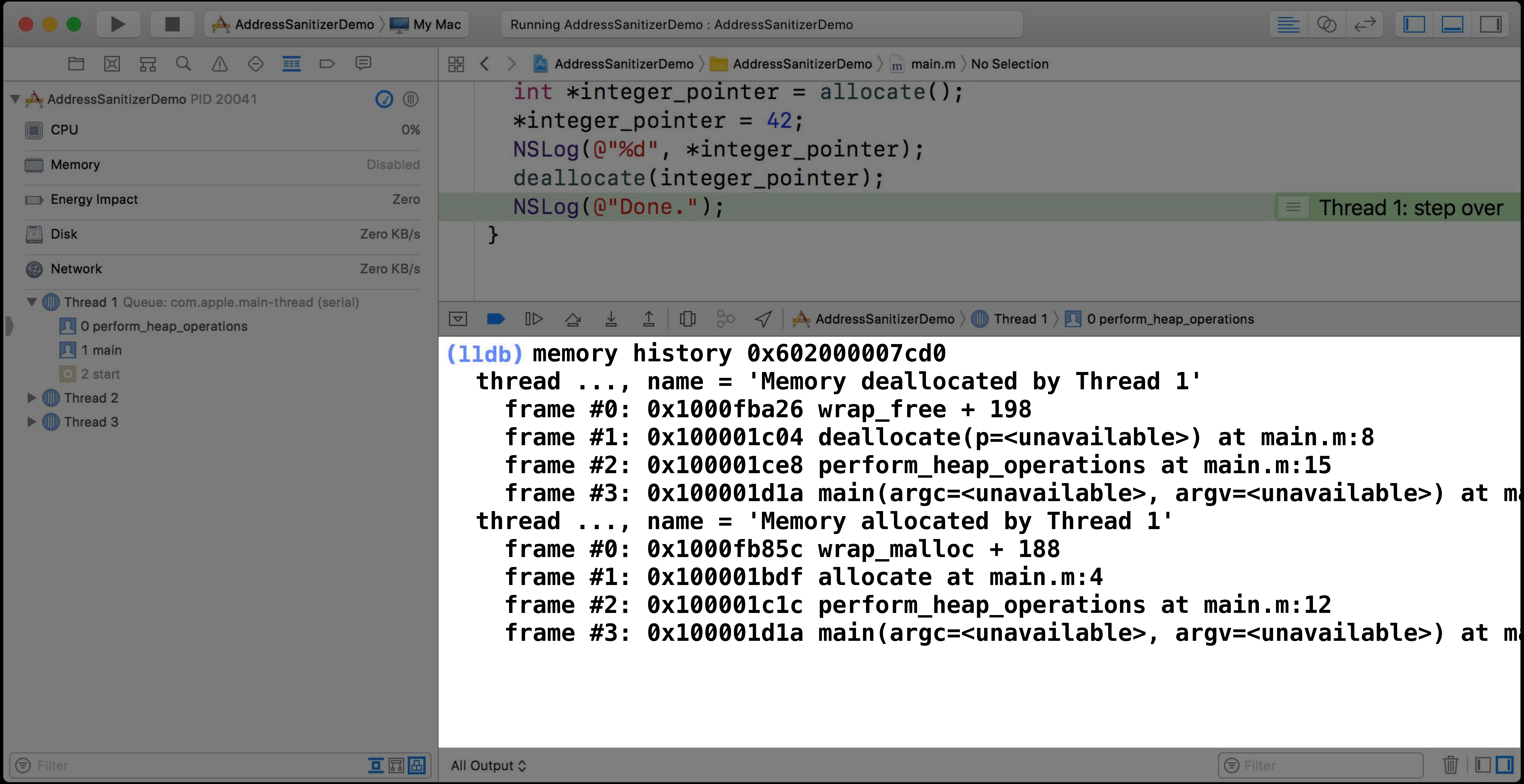
Thread 1: step over

```
(lldb) memory history <expression>
```



```
    int *integer_pointer = allocate();
    *integer_pointer = 42;
    NSLog(@"%d", *integer_pointer);
    deallocate(integer_pointer);
    NSLog(@"Done.");
}
```

```
(lldb) memory history 0x602000007cd0
```



```
    int *integer_pointer = allocate();
    *integer_pointer = 42;
    NSLog(@"%d", *integer_pointer);
    deallocate(integer_pointer);
    NSLog(@"Done.");
}
```

```
(lldb) memory history 0x602000007cd0
  thread ..., name = 'Memory deallocated by Thread 1'
    frame #0: 0x1000fba26 wrap_free + 198
    frame #1: 0x100001c04 deallocate(p=<unavailable>) at main.m:8
    frame #2: 0x100001ce8 perform_heap_operations at main.m:15
    frame #3: 0x100001d1a main(argc=<unavailable>, argv=<unavailable>) at ma
  thread ..., name = 'Memory allocated by Thread 1'
    frame #0: 0x1000fb85c wrap_malloc + 188
    frame #1: 0x100001bdf allocate at main.m:4
    frame #2: 0x100001c1c perform_heap_operations at main.m:12
    frame #3: 0x100001d1a main(argc=<unavailable>, argv=<unavailable>) at ma
```

When to Use Address Sanitizer

C languages and Swift

Memory corruptions and crashes

General debugging

Thread Sanitizer

Detects multithreading problems

What is Thread Sanitizer

Multithreading issues

Finds races even if they did not manifest

64-bit macOS, 64-bit simulators

Data Races

Unsynchronized accesses to shared mutable variables

Lead to data races

Memory corruptions and crashes

All of these problems apply to Swift!

```swift
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

✕

```swift
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

```swift
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

```swift
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

```swift
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

```swift
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}

// Thread 1
eventLog.log(source: networkingSubsystem, message: "Download finished")

// Thread 2
eventLog.log(source: databaseSubsystem, message: "Query complete")
```

```swift
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}


// Thread 1
eventLog.log(source: networkingSubsystem, message: "Download finished")

// Thread 2
eventLog.log(source: databaseSubsystem, message: "Query complete")
```

```
// Swift Data Race Example

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}


// Thread 1
eventLog.log(source: networkingSubsystem, message: "Download finished")

// Thread 2
eventLog.log(source: databaseSubsystem, message: "Query complete")
```

! Thread 2: Data race in EventLog.log(source:message:)

```swift
// Use DispatchQueue to Synchronize Access

class EventLog {
  private var lastEventSource: LogSource?

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

```swift
// Use DispatchQueue to Synchronize Access

class EventLog {
  private var lastEventSource: LogSource?
  private var queue = DispatchQueue(label: "com.example.EventLog.queue")

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

```
// Use DispatchQueue to Synchronize Access

class EventLog {
  private var lastEventSource: LogSource?
  private var queue = DispatchQueue(label: "com.example.EventLog.queue")

  func log(source: LogSource, message: String) {
    print(message)
    lastEventSource = source
  }
}
```

✓

```swift
// Use DispatchQueue to Synchronize Access

class EventLog {
  private var lastEventSource: LogSource?
  private var queue = DispatchQueue(label: "com.example.EventLog.queue")

  func log(source: LogSource, message: String) {
    queue.async {
      print(message)
      lastEventSource = source
    }
  }
}
```

✓

```swift
// Use DispatchQueue to Synchronize Access

class EventLog {
  private var lastEventSource: LogSource?
  private var queue = DispatchQueue(label: "com.example.EventLog.queue")

  func log(source: LogSource, message: String) {
    queue.async {
      print(message)
      lastEventSource = source
    }
  }
}
```

✓

```
// Use DispatchQueue to Synchronize Access

class EventLog {
  private var lastEventSource: LogSource?
  private var queue = DispatchQueue(label: "com.example.EventLog.queue")

  func log(source: LogSource, message: String) {
    queue.async {
      print(message)
      lastEventSource = source
    }
  }
}
```

Dispatch Queues

Grand Central Dispatch should be your first choice of synchronization

Lightweight, convenient, simple

Associate your data with serial dispatch queues

New in Thread Sanitizer in Xcode 9

NEW

Races on collections

Swift access races

Races on Collections

Previously, only reported races on raw memory accesses

Synchronization required for larger data structures

Races on Collections

Previously, only reported races on raw memory accesses

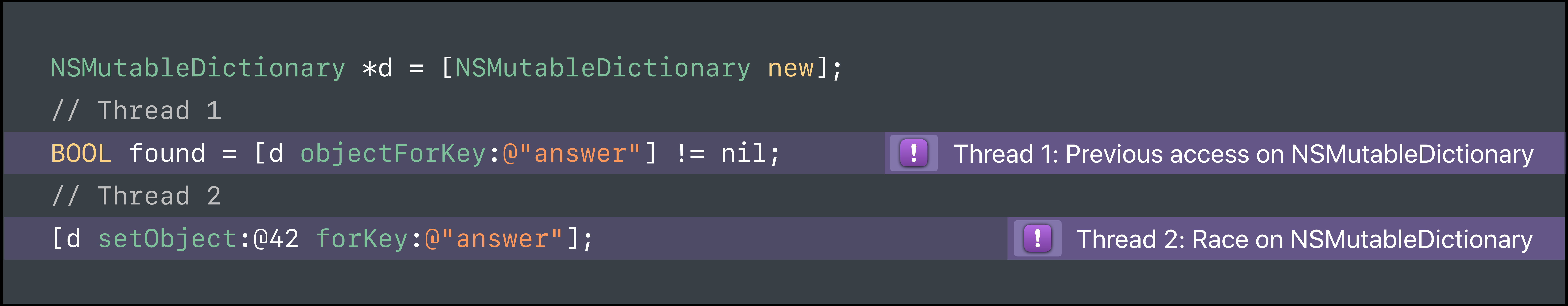
Synchronization required for larger data structures

```
NSMutableDictionary *d = [NSMutableDictionary new];
// Thread 1
BOOL found = [d objectForKey:@"answer"] != nil;
// Thread 2
[d setObject:@42 forKey:@"answer"];
```

Races on Collections

Previously, only reported races on raw memory accesses

Synchronization required for larger data structures



```
NSMutableDictionary *d = [NSMutableDictionary new];
// Thread 1
BOOL found = [d objectForKey:@"answer"] != nil;
// Thread 2
[d setObject:@42 forKey:@"answer"];
```

! Thread 1: Previous access on NSMutableDictionary

! Thread 2: Race on NSMutableDictionary

Races on Collections

NEW

Races on collections in Objective-C and Swift

`NSMutableArray`, `NSMutableDictionary`

Swift `Array` and `Dictionary`

Demo

Thread Sanitizer and race on NSMutableArray

```
// Race on Swift Array

var usernames: [String] = ["alice", "bob"]
```

✕

```swift
// Race on Swift Array

var usernames: [String] = ["alice", "bob"]

// Thread 1
found = usernames.contains("alice")
if found { … }

// Thread 2
usernames.append("carol")
```

✗

```
// Race on Swift Array

var usernames: [String] = ["alice", "bob"]

// Thread 1
found = usernames.contains("alice")                    ! Thread 1: Previous access
if found { … }

// Thread 2
usernames.append("carol")                              ! Thread 2: Swift access race
```

```swift
// Use DispatchQueue to Synchronize Accesses

var usernames: [String] = ["alice", "bob"]

// Thread 1
found = usernames.contains("alice")
if found { … }

// Thread 2
usernames.append("carol")
```

```swift
// Use DispatchQueue to Synchronize Accesses

var usernames: [String] = ["alice", "bob"]
var queue = DispatchQueue(label: "com.example.usernames.queue")

// Thread 1
found = usernames.contains("alice")
if found { … }

// Thread 2
usernames.append("carol")
```

```swift
// Use DispatchQueue to Synchronize Accesses

var usernames: [String] = ["alice", "bob"]
var queue = DispatchQueue(label: "com.example.usernames.queue")

// Thread 1
found = usernames.contains("alice")
if found { … }

// Thread 2
usernames.append("carol")
```

```swift
// Use DispatchQueue to Synchronize Accesses

var usernames: [String] = ["alice", "bob"]
var queue = DispatchQueue(label: "com.example.usernames.queue")

// Thread 1
found = usernames.contains("alice")
if found { … }

// Thread 2
usernames.append("carol")
```

✓

```
// Use DispatchQueue to Synchronize Accesses

var usernames: [String] = ["alice", "bob"]
var queue = DispatchQueue(label: "com.example.usernames.queue")

// Thread 1
queue.sync {
  found = usernames.contains("alice")
}
if found { … }

// Thread 2
queue.async {
  usernames.append("carol")
}
```

✓

```swift
// Use DispatchQueue to Synchronize Accesses

var usernames: [String] = ["alice", "bob"]
var queue = DispatchQueue(label: "com.example.usernames.queue")

// Thread 1
queue.sync {
  found = usernames.contains("alice")
}
if found { … }

// Thread 2
queue.async {
  usernames.append("carol")
}
```

Swift Access Races

NEW

Applies to all structs

Mutating methods require exclusive access to the whole struct

Methods on classes require exclusive access to stored properties they change

```
// Swift Access Race with Mutating Methods

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int




}
```

✗

```swift
// Swift Access Race with Mutating Methods

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int

  mutating func teleport(toPlanet: String) { … }
  mutating func fly(toCity: String) { … }
  mutating func travelToEndOfTime() { … }
}
```

```swift
// Swift Access Race with Mutating Methods

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int

  mutating func teleport(toPlanet: String) { … }
  mutating func fly(toCity: String) { … }
  mutating func travelToEndOfTime() { … }
}
```

✕

```swift
// Swift Access Race with Mutating Methods

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int

  mutating func teleport(toPlanet: String) { … }
  mutating func fly(toCity: String) { … }
  mutating func travelToEndOfTime() { … }
}

// Thread 1
location.teleport(toPlanet: "Mars")

// Thread 2
location.travelToEndOfTime()
```

```swift
// Swift Access Race with Mutating Methods

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int

  mutating func teleport(toPlanet: String) { … }
  mutating func fly(toCity: String) { … }
  mutating func travelToEndOfTime() { … }
}

// Thread 1
location.teleport(toPlanet: "Mars")                    ! Thread 1: Previous access

// Thread 2
location.travelToEndOfTime()                           ! Thread 2: Swift access race
```

✕

```
// Swift Access Race with Mutating Methods

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int

  mutating func teleport(toPlanet: String) { … }
  mutating func fly(toCity: String) { … }
  mutating func travelToEndOfTime() { … }
}

// Thread 1
location.teleport(toPlanet: "Mars")

// Thread 2
location.travelToEndOfTime()
```

changes x, y, z

changes time

! Thread 1: Previous access

! Thread 2: Swift access race

✕

```
// Swift Access Race with Mutating Methods

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int

  mutating func teleport(toPlanet: String) { … }
  mutating func fly(toCity: String) { … }
  mutating func travelToEndOfTime() { … }
}

// Thread 1
location.teleport(toPlanet: "Mars")                    ! Thread 1: Previous access

// Thread 2
location.travelToEndOfTime()                           ! Thread 2: Swift access race
```

```swift
// Incorrect Synchronization Inside a Struct

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int

  mutating func teleport(toPlanet: String) { … }
  mutating func fly(toCity: String) { … }
  mutating func travelToEndOfTime() { … }
}
```

```swift
// Incorrect Synchronization Inside a Struct

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int
  private var queue: DispatchQueue = …

  mutating func teleport(toPlanet: String) { queue.sync { … } }
  mutating func fly(toCity: String) { queue.sync { … } }
  mutating func travelToEndOfTime() { queue.sync { … } }
}
```

```swift
// Incorrect Synchronization Inside a Struct

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int
  private var queue: DispatchQueue = …

  mutating func teleport(toPlanet: String) { queue.sync { … } }
  mutating func fly(toCity: String) { queue.sync { … } }
  mutating func travelToEndOfTime() { queue.sync { … } }
}
```

```swift
// Incorrect Synchronization Inside a Struct

struct BluePoliceBoxLocation {
  private var x, y, z: Int
  private var time: Int
  private var queue: DispatchQueue = …

  mutating func teleport(toPlanet: String) { queue.sync { … } }
  mutating func fly(toCity: String) { queue.sync { … } }
  mutating func travelToEndOfTime() { queue.sync { … } }
}
```

```swift
// Synchronize Calls to Mutating Methods

struct BluePoliceBoxLocation { … }
```

✓

✓

```
// Synchronize Calls to Mutating Methods

struct BluePoliceBoxLocation { … }
class BluePoliceBox {
  private var location: BluePoliceBoxLocation
  private var queue: DispatchQueue = …

}
```

✓

```
// Synchronize Calls to Mutating Methods

struct BluePoliceBoxLocation { … }
class BluePoliceBox {
  private var location: BluePoliceBoxLocation
  private var queue: DispatchQueue = …
  func goOnRescueMission() {
    queue.sync {
      location.teleport(toPlanet: "Mars")
      …
    }
  }
  func goToWrongPlaceAgain() {
    queue.sync {
      …
    }
  }
}
```

Find and Fix Your Races

Use GCD to synchronize access to data

Associate your shared data with a serial queue

Thread Sanitizer is invaluable for finding races

NEW

Undefined Behavior Sanitizer

Vedant Kumar, Compiler Engineer

What is Undefined Behavior Sanitizer?

Runtime bug finder

Checks unsafe constructs in the C language family

Compatible with other runtime tools

Understanding Undefined Behavior   Executive Ballroom   Thursday 9:00AM

C++ Dynamic Type Violation

Invalid Float Cast

Nonnull Return Value Violation

Integer Overflow

Invalid Shift Exponent

Alignment Violation

Invalid Boolean

Invalid Variable-Length Array

Invalid Enum

Integer Division by Zero

Invalid Integer Cast

Reached Unreachable Code

Invalid Shift Base

Missing Return Value

Invalid Object Size

Null Dereference

Nonnull Assignment Violation

Nonnull Parameter Violation

Out-of-Bounds Array Access

Integer Overflow

Alignment Violation

Nonnull Return Value Violation

Integer Overflow

Arithmetic result too big

Unsafe in indexing expressions

```
(INT_MAX + 1) ≯ INT_MAX
```

Opt-in check for unsigned overflow

Demo

Undefined Behavior Sanitizer and integer overflow

Alignment Violation

Unaligned load or store

Causes crashes in Release builds

Common in (de)serialization code

```
// Serializing Packets for a Custom Network Protocol

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Serializing Packets for a Custom Network Protocol
```

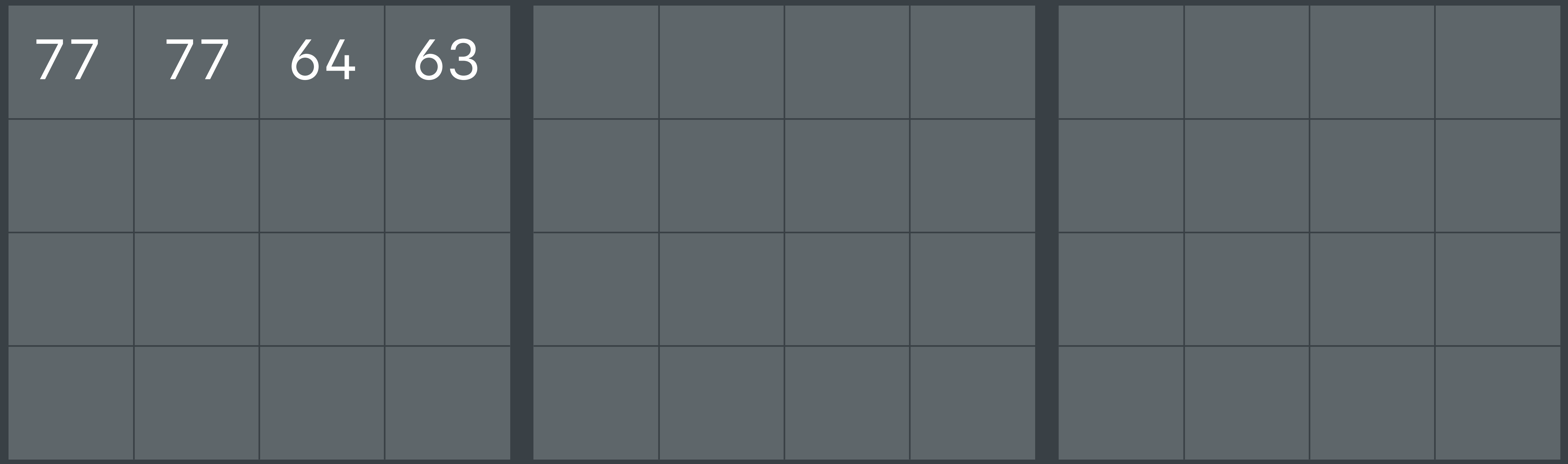
```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```

```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

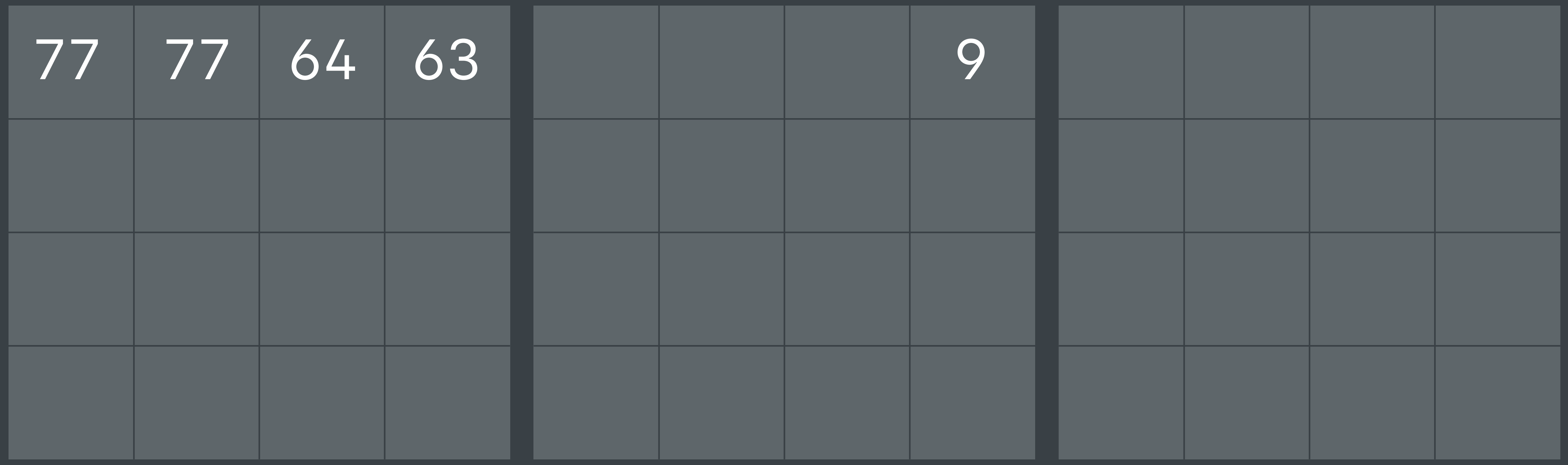
```
// Sender
```



```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```

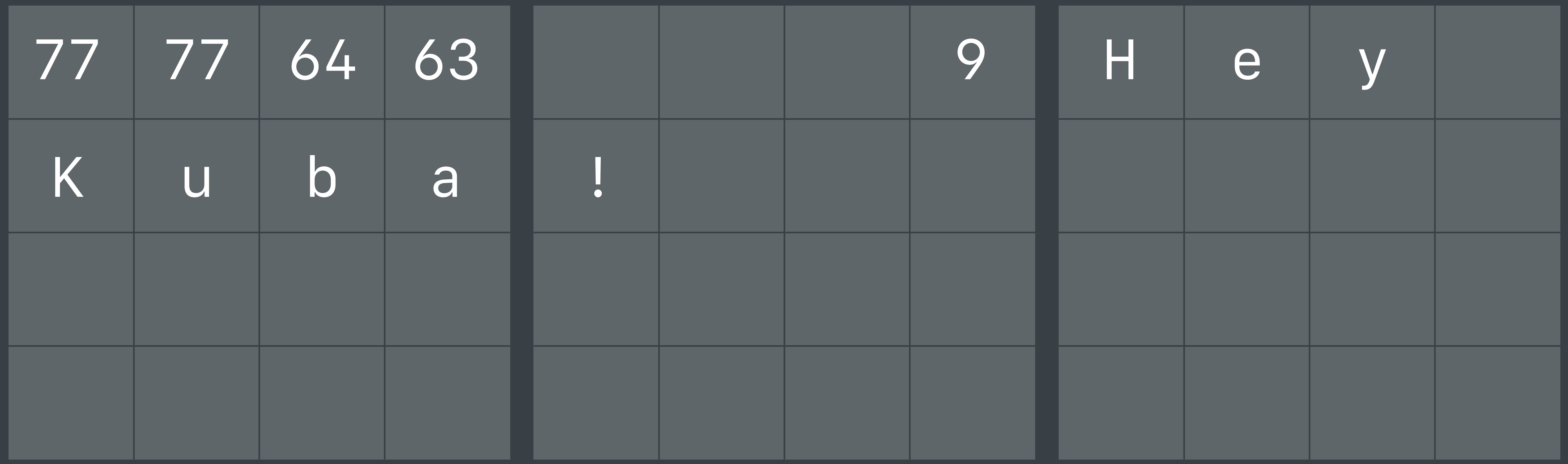


77	77	64	63				9				

```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```

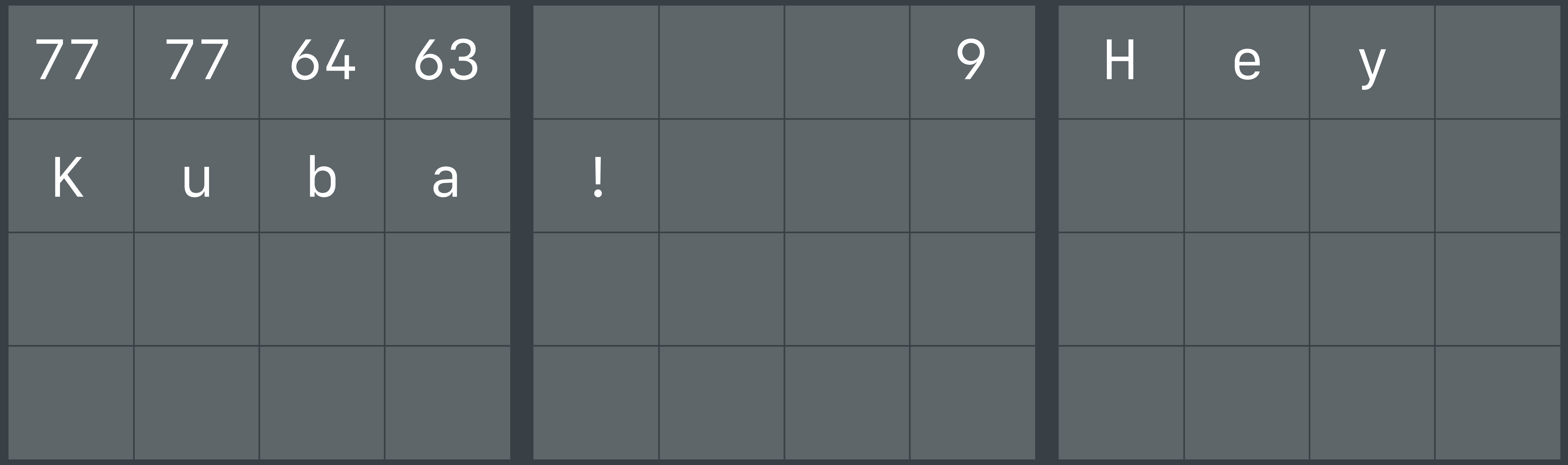


77	77	64	63				9	H	e	y	
K	u	b	a	!							

```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```



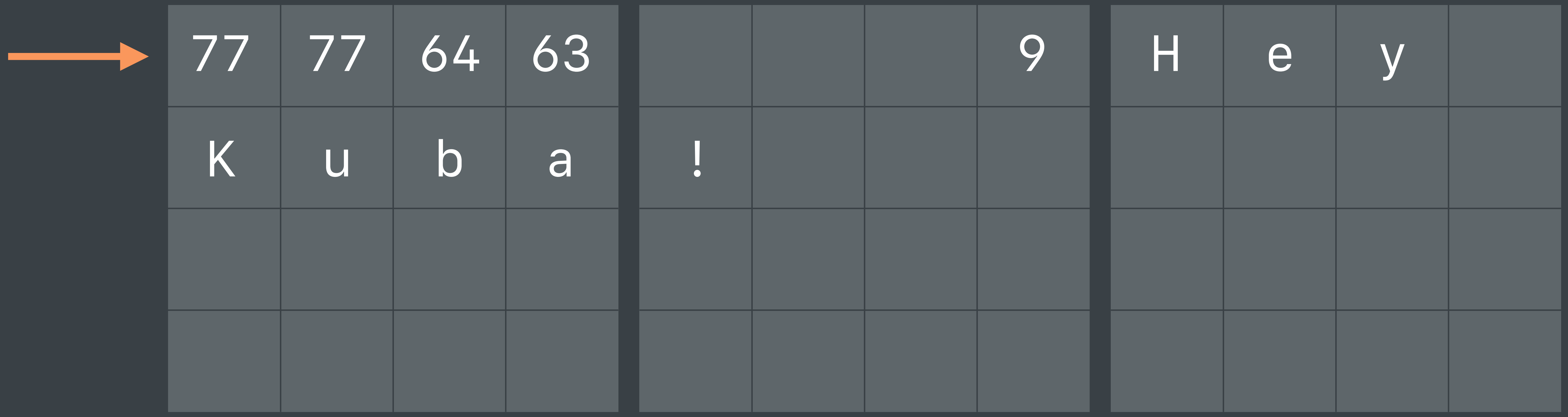
77	77	64	63				9	H	e	y	
K	u	b	a	!							

```
// Receiver
// Read from stream
Packet *P = (Packet *)byteStream;
if (P->magic != …)
    …
```

```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```



77	77	64	63				9	H	e	y	
K	u	b	a	!							

```
// Receiver
// Read from stream
Packet *P = (Packet *)byteStream;
if (P->magic != …)
    …
```

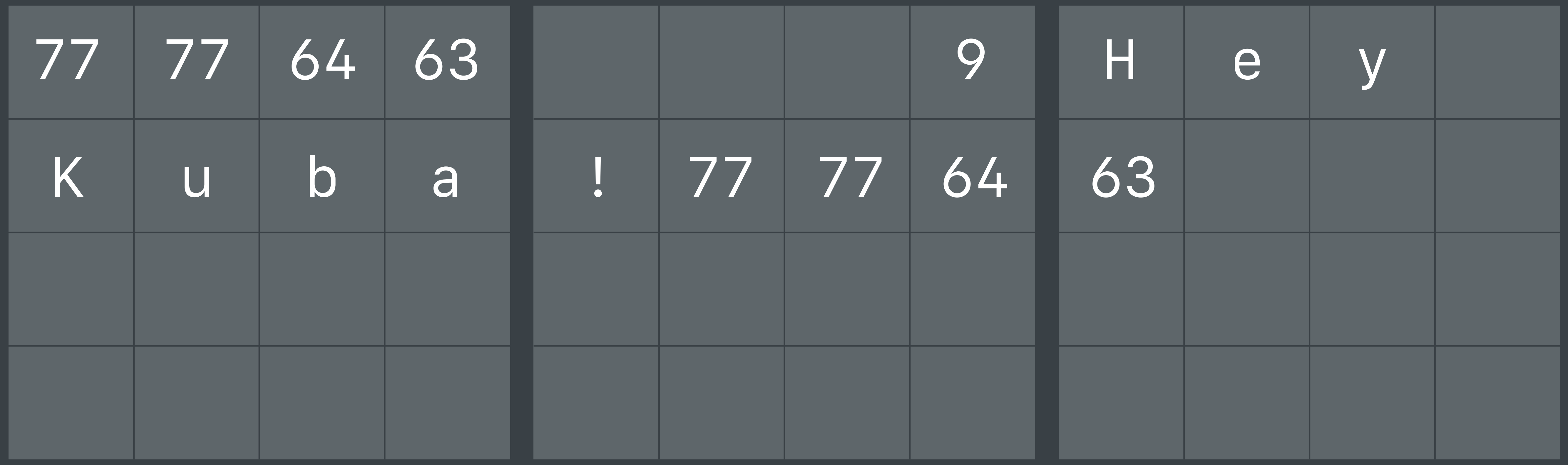
```
// Serializing Packets for a Custom Network Protocol

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```

77	77	64	63				9	H	e	y	
K	u	b	a	!							

```
// Serializing Packets for a Custom Network Protocol

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

// Sender



77	77	64	63				9	H	e	y	
K	u	b	a	!	77	77	64	63			

```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```

77	77	64	63				9	H	e	y	
K	u	b	a	!	77	77	64	63			
15											

```
// Serializing Packets for a Custom Network Protocol

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```

77	77	64	63
K	u	b	a
15	H	o	w
i	n	g	?

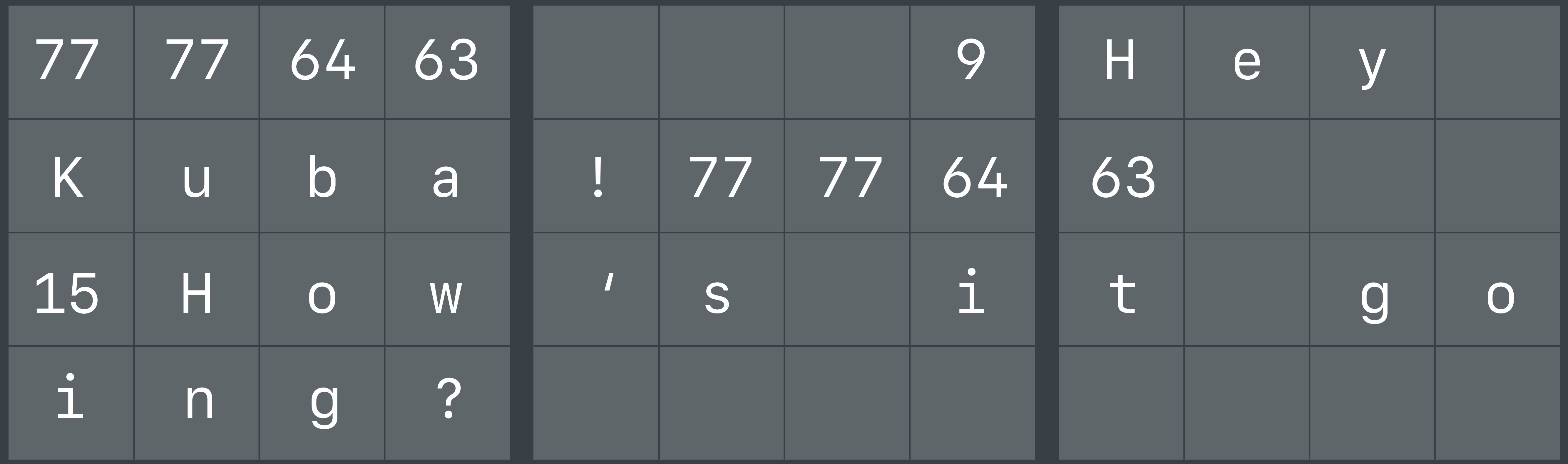
			9
!	77	77	64
'	s		i

H	e	y	
63			
t		g	o

```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```



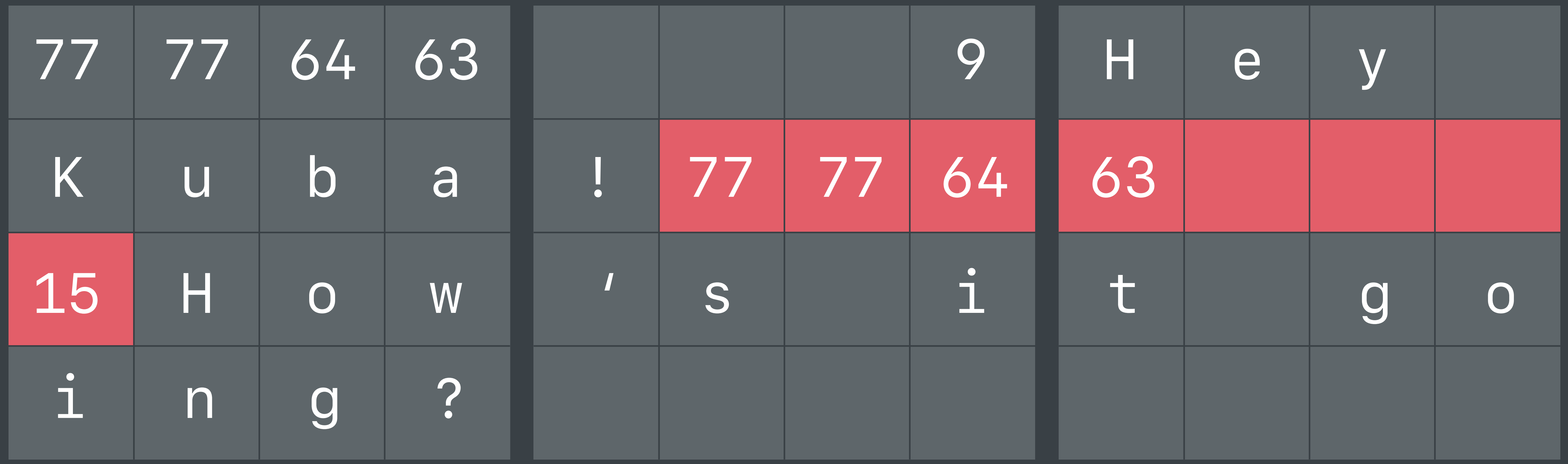
77	77	64	63				9	H	e	y	
K	u	b	a	!	77	77	64	63			
15	H	o	w	’	s		i	t		g	o
i	n	g	?								

```
// Receiver
// Read from stream
Packet *P = (Packet *)(byteStream + 17);
if (P->magic != …)
    …
```

```
// Serializing Packets for a Custom Network Protocol
```

```
struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```



77	77	64	63				9	H	e	y	
K	u	b	a	!	77	77	64	63			
15	H	o	w	'	s		i	t		g	o
i	n	g	?								

```
// Receiver
// Read from stream
Packet *P = (Packet *)(byteStream + 17);
if (P->magic != …)
    …
```

```
// Serializing Packets for a Custom Network Protocol

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```

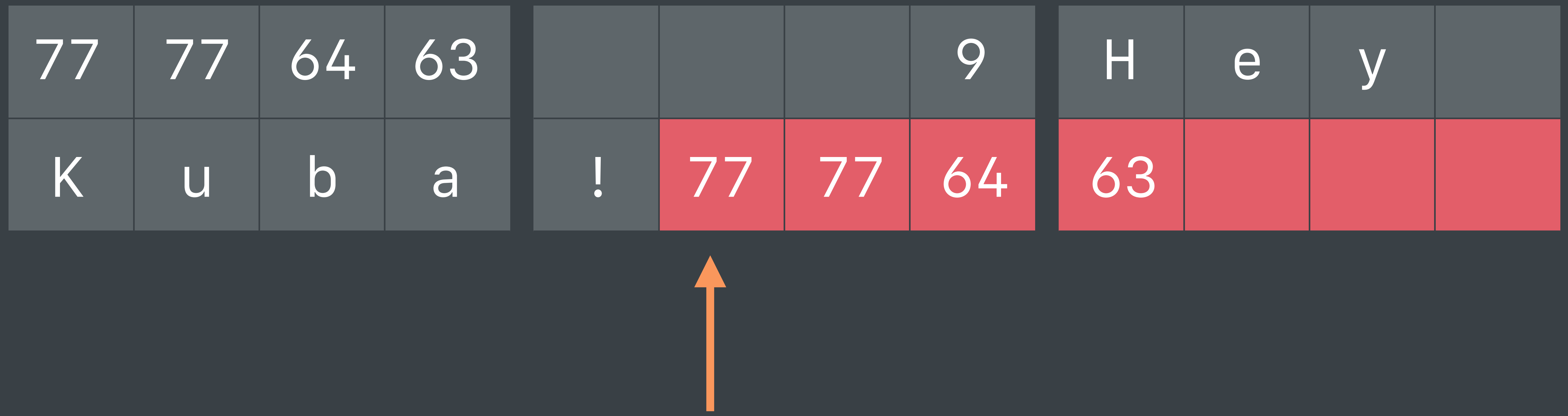
77	77	64	63				9	H	e	y	
K	u	b	a	!	77	77	64	63			

```
// Receiver
// Read from stream
Packet *P = (Packet *)(byteStream + 17);
if (P->magic != …)
    …
```

! Load of misaligned address

```
// Serializing Packets for a Custom Network Protocol

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};
```

```
// Sender
```



77	77	64	63				9	H	e	y	
K	u	b	a	!	77	77	64	63			

```
// Receiver
// Read from stream
Packet *P = (Packet *)(byteStream + 17);
if (P->magic != …)
    …
```

❗ Load of misaligned address

✓

```
// Use Structure Packing to Decrease Expected Alignment

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
} __attribute__((packed));

// Read from stream
Packet *P = (Packet *)byteStream;
if (P->magic != …)
    …
```

✓

```
// Use Structure Packing to Decrease Expected Alignment

struct Packet {
    int magic; // Member alignment changes to 1
    int payloadLength;
    char payload[];
} __attribute__((packed));

// Read from stream
Packet *P = (Packet *)byteStream;
if (P->magic != …)
    …
```

✓

```
// Use Structure Packing to Decrease Expected Alignment

struct Packet {
    int magic; // Member alignment changes to 1
    int payloadLength;
    char payload[];
} __attribute__((packed));

// Read from stream
Packet *P = (Packet *)byteStream;
if (P->magic != …) // The load is aligned
    …
```

✓

```
// Use Structure Packing to Decrease Expected Alignment

struct Packet {
    int magic; // Member alignment changes to 1
    int payloadLength;
    char payload[];
} __attribute__((packed)); // This can change structure layout and performance

// Read from stream
Packet *P = (Packet *)byteStream;
if (P->magic != …)  // The load is aligned
    …
```

✓

```
// Use memcpy() to Perform Unaligned Accesses

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};

// Read from stream
int magic;
memcpy(&magic, byteStream + offsetof(struct Packet, magic), sizeof(int));
if (magic != …)
    …
```

✓

```
// Use memcpy() to Perform Unaligned Accesses

struct Packet {
    int magic;
    int payloadLength;
    char payload[];
};

// Read from stream
int magic;
memcpy(&magic, byteStream + offsetof(struct Packet, magic), sizeof(int));
if (magic != …)
    …
```

Nonnull Return Value Violation

Return value annotated `nonnull`

Function returns `nil` anyway

Can cause crashes in mixed C and Swift code

Recommended to opt in to the check

```
// Nonnull Return Value Violation

@implementation SolarSystem
+ (nonnull NSDictionary *)planetMoons {
  return @{@"Earth": @[@"Moon"],
           @"Mars" : @[@"Phobos", @"Deimos"],
           // …
           @"Pluto": @[@"Charon", @"Hydra", @"Nix", @"Kerberos", @"Styx"]
          };
}
- (nonnull NSArray *)moonsOfPlanet:(nonnull NSString *)planet {
  return [[self class] planetMoons][planet];
}
@end
```

```
// Nonnull Return Value Violation

@implementation SolarSystem
+ (nonnull NSDictionary *)planetMoons {
  return @{@"Earth": @[@"Moon"],
           @"Mars" : @[@"Phobos", @"Deimos"],
           // …
           @"Pluto": @[@"Charon", @"Hydra", @"Nix", @"Kerberos", @"Styx"]
          };
}
- (nonnull NSArray *)moonsOfPlanet:(nonnull NSString *)planet {
  return [[self class] planetMoons][planet];
}
@end
```

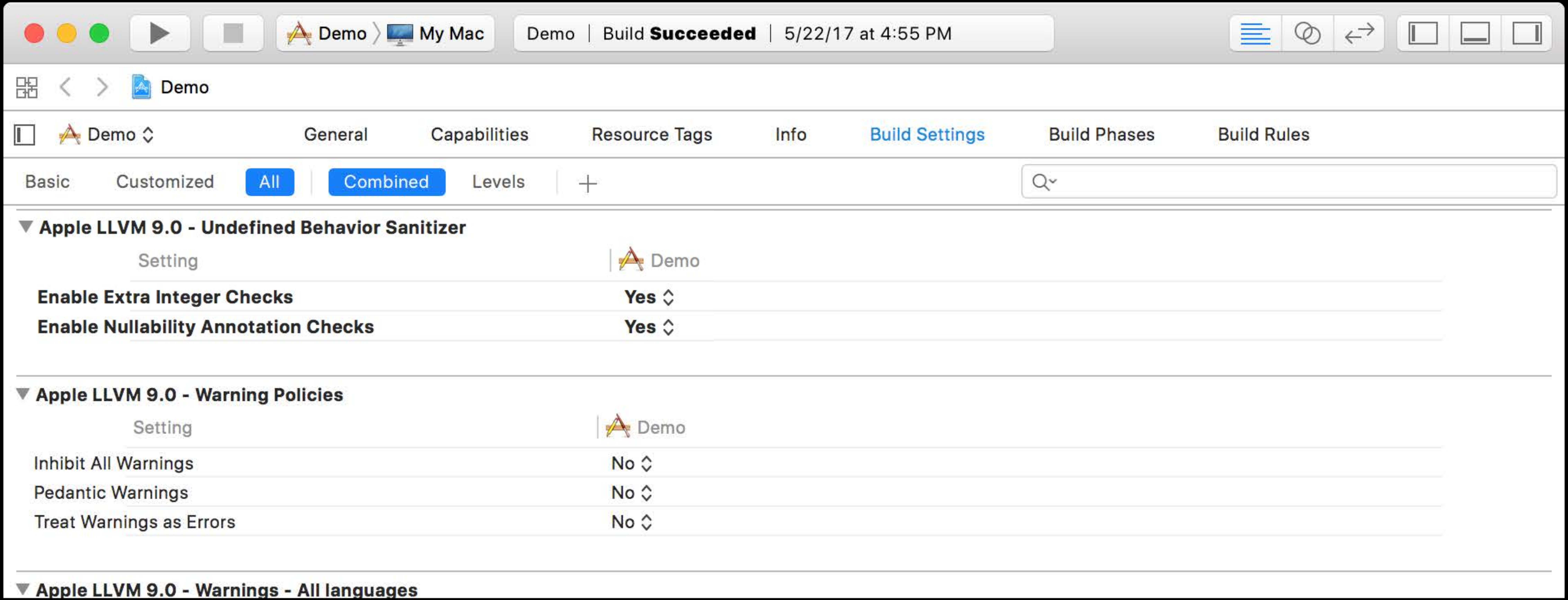
```
// Nonnull Return Value Violation

@implementation SolarSystem
+ (nonnull NSDictionary *)planetMoons {
  return @{@"Earth": @[@"Moon"],
           @"Mars" : @[@"Phobos", @"Deimos"],
           // …
          };
}
- (nonnull NSArray *)moonsOfPlanet:(nonnull NSString *)planet {
  return [[self class] planetMoons][planet];
}
@end
```

```
// Nonnull Return Value Violation

@implementation SolarSystem
+ (nonnull NSDictionary *)planetMoons {
  return @{@"Earth": @[@"Moon"],
           @"Mars" : @[@"Phobos", @"Deimos"],
           // …
          };
}
- (nonnull NSArray *)moonsOfPlanet:(nonnull NSString *)planet {
  return [[self class] planetMoons][planet];
}
@end
```

```
// Nonnull Return Value Violation

@implementation SolarSystem
+ (nonnull NSDictionary *)planetMoons {
  return @{@"Earth": @[@"Moon"],
           @"Mars" : @[@"Phobos", @"Deimos"],
           // …
          };
}
- (nonnull NSArray *)moonsOfPlanet:(nonnull NSString *)planet {
  return [[self class] planetMoons][planet];
}
@end

// Find the biggest moon for each planet
NSMutableArray *biggestMoons = [NSMutableArray new];
[biggestMoons addObject:[solarSystem moonsOfPlanet:@"Pluto"][0]];
```

✕

```
// Nonnull Return Value Violation

@implementation SolarSystem
+ (nonnull NSDictionary *)planetMoons {
  return @{@"Earth": @[@"Moon"],
           @"Mars" : @[@"Phobos", @"Deimos"],
           // …
          };
}
- (nonnull NSArray *)moonsOfPlanet:(nonnull NSString *)planet {
  return [[self class] planetMoons][planet];  ! Null pointer returned from function declared to never return null
}
@end

// Find the biggest moon for each planet
NSMutableArray *biggestMoons = [NSMutableArray new];
[biggestMoons addObject:[solarSystem moonsOfPlanet:@"Pluto"][0]];
```



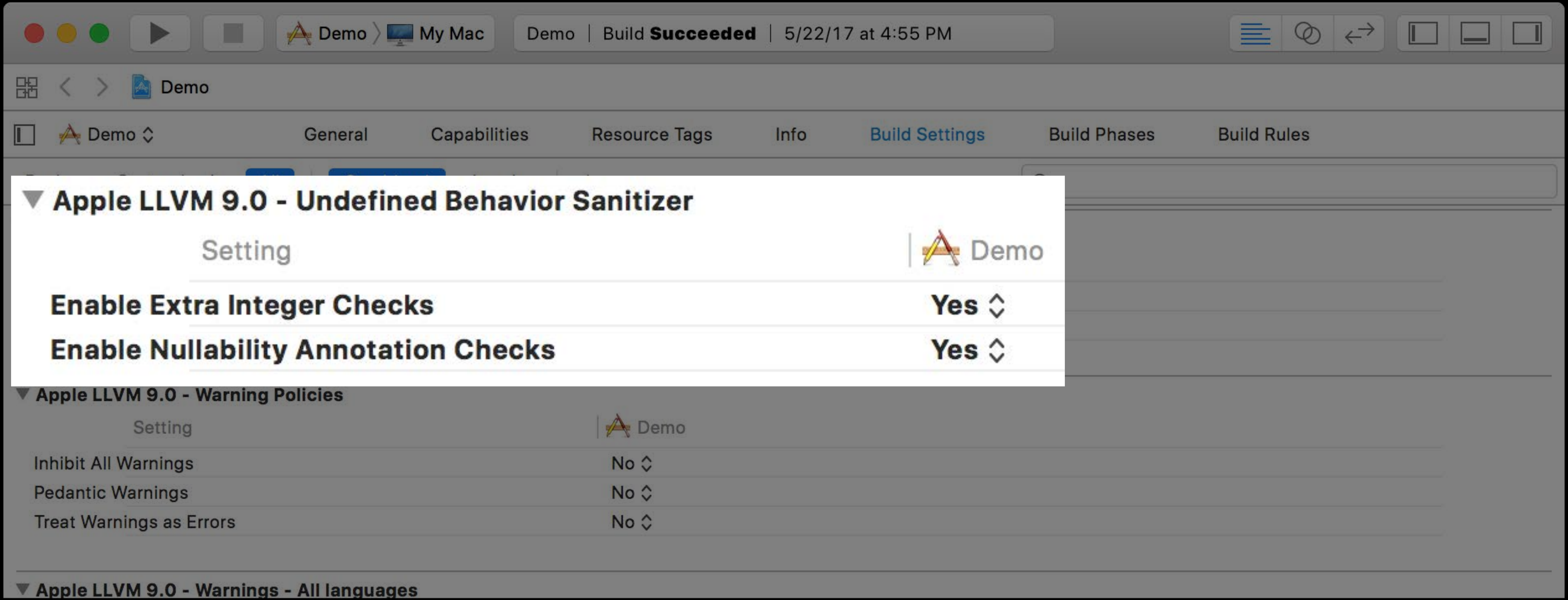
Apple LLVM 9.0 - Undefined Behavior Sanitizer

Setting	Demo
Enable Extra Integer Checks	**Yes**
Enable Nullability Annotation Checks	**Yes**

Apple LLVM 9.0 - Warning Policies

Setting	Demo
Inhibit All Warnings	No
Pedantic Warnings	No
Treat Warnings as Errors	No

Apple LLVM 9.0 - Warnings - All languages



Apple LLVM 9.0 - Undefined Behavior Sanitizer

Setting	Demo
Enable Extra Integer Checks	Yes
Enable Nullability Annotation Checks	Yes

Apple LLVM 9.0 - Warning Policies

Setting	Demo
Inhibit All Warnings	No
Pedantic Warnings	No
Treat Warnings as Errors	No

Apple LLVM 9.0 - Warnings - All languages

Using Runtime Tools Effectively

How to Use Runtime Tools Effectively

Exercise more code

Use the tools together

Exercise More Code

Can only catch issues in code that is run

Use runtime tools for daily development

Use them before every release

Avoid spreading bugs to users

Use Continuous Integration

Simplifies testing with runtime tools

Ensures that bugs are caught quickly

Helps track code coverage

Use Runtime Tools Together

Find more issues

Most runtime tools can be used together

- Address Sanitizer and Thread Sanitizer are not compatible

Product → Scheme → Edit Scheme… → Diagnostics

Runtime Tool Overhead

	Execution overhead	Memory overhead
Main Thread Checker	1.02x	negligible
Undefined Behavior Sanitizer	1.2x	negligible
Address Sanitizer	2–3x	2x
Thread Sanitizer	5–10x	4x

Summary

Xcode 9 enables you to catch critical issues

Use runtime tools early and often

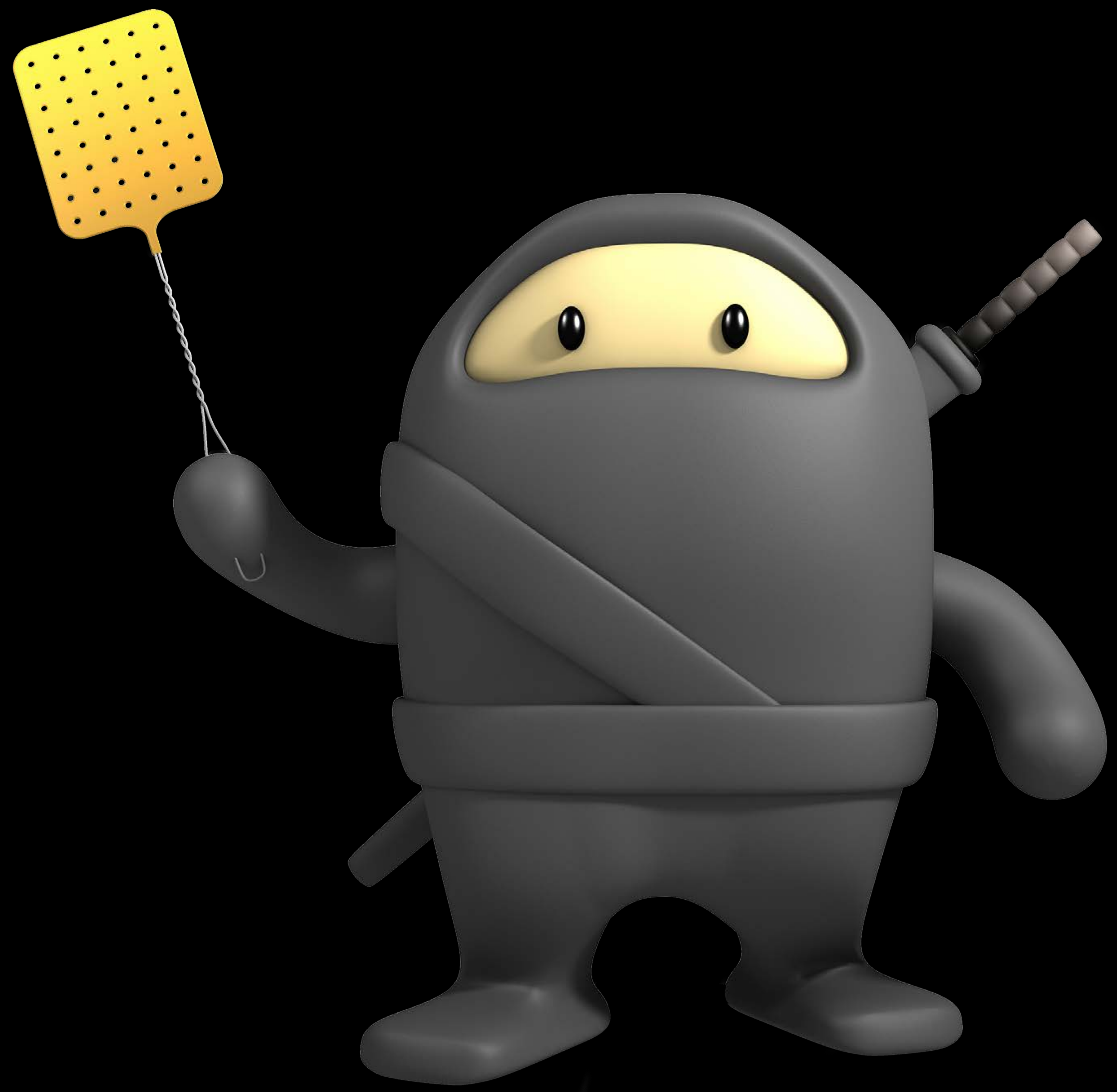
Save time, keep users safe!

Summary

Xcode 9 enables you to catch critical issues

Use runtime tools early and often

Save time, keep users safe!



More Information

https://developer.apple.com/wwdc17/406

Related Sessions

What's New in Swift		Tuesday 1:50PM
Debugging with Xcode 9		Wednesday 10:00AM
Modernizing Grand Central Dispatch Usage		Wednesday 11:00AM
Understanding Undefined Behavior	Executive Ballroom	Thursday 9:00AM
What’s New in Testing	Hall 2	Thursday 3:10PM
What's New in LLVM	Hall 2	Thursday 4:10PM

Labs

Performance Profiling and Runtime Analysis Tools Lab	Technology Lab K	Thu 1:00PM–4:10PM
LLVM Compiler, Objective-C, and C++ Lab	Technology Lab E	Fri 9:00AM–11:00AM

WWDC17