
Google™

# Crashing Harder

## Catching your application in compromising contexts

chrome

**Mark Mentovai**
Software Engineer, Google Chrome

# What Is a Crash?

Google

## Hardware traps

xnu-2050.18.24/osfmk/i386/trap.h (10.8.2, paraphrased):

```
#define T_DIVIDE_ERROR        0
#define T_INT3                3   /* int 3 instruction */
#define T_INVALID_OPCODE      6   /* invalid op code */
#define T_GENERAL_PROTECTION  13
#define T_PAGE_FAULT          14
```

# What Is a Crash?

Google

## Mach exceptions

/usr/include/mach/exception_types.h (10.8.2, paraphrased):

```
/*
 *      Machine-independent exception definitions.
 */

#define EXC_BAD_ACCESS       1   /* Could not access memory */
             /* Code contains kern_return_t describing error. */
             /* Subcode contains bad memory address. */

#define EXC_BAD_INSTRUCTION  2   /* Instruction failed */
             /* Illegal or undefined instruction or operand */

#define EXC_ARITHMETIC       3   /* Arithmetic exception */
             /* Exact nature of exception is in code field *

#define EXC_BREAKPOINT       6  /* Trace, breakpoint, etc. */
             /* Details in code field. */

#define EXC_CRASH            10  /* Abnormal process exit */
```

# What Is a Crash?

Google

## UNIX Signals

/usr/include/sys/signal.h (10.8.2, paraphrased):

```
#define SIGILL  4   /* illegal instruction (not reset when
                       caught) */
#define SIGTRAP 5   /* trace trap (not reset when caught) */
#define SIGABRT 6   /* abort() */
#define SIGFPE  8   /* floating point exception */
#define SIGBUS  10  /* bus error */
#define SIGSEGV 11  /* segmentation violation */
```

# What Is a Crash?

Google

## Trap T_DIVIDE_ERROR: Integer division by zero

```
int main(int argc, char* argv[]) {
  int dividend = 1;
  int divisor = 0;
  int quotient = dividend / divisor;
  return 0;
}
```

```
bash$ ./trap_0_divide_error
Floating point exception: 8

Exception Type:  EXC_ARITHMETIC (SIGFPE)
Exception Codes: EXC_I386_DIV (divide by zero)
```

# What Is a Crash?

Google

## Trap `T_INVALID_OPCODE`: Invalid opcode

```
int main(int argc, char* argv[]) {
  __builtin_trap();
  return 0;
}
```

```
bash$ ./trap_6_invalid_opcode
Illegal instruction: 4

Exception Type:  EXC_BAD_INSTRUCTION (SIGILL)
Exception Codes: 0x0000000000000001, 0x0000000000000000
```

# What Is a Crash?

Google

Trap T_GENERAL_PROTECTION: General protection fault

```
#include <stddef.h>

int main(int argc, char* argv[]) {
  int* pointer = NULL;
  *pointer = 6;
  return 0;
}
```

```
bash$ ./trap_13_general_protection
Bus error: 10

Exception Type:  EXC_BAD_ACCESS (SIGBUS)
Exception Codes: KERN_PROTECTION_FAILURE at 0x0000000000000000
```

# What Is a Crash?

Google

Software: `abort()`

```
#include <stdlib.h>

int main(int argc, char* argv[]) {
  abort();
  return 0;
}
```

```
bash$ ./abort
Abort trap: 6

Exception Type:  EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Application Specific Information:
abort() called
```

# What Is a Crash?

Google

Software: `assert()`

```
#include <assert.h>

int main(int argc, char* argv[]) {
  assert(false);
  return 0;
}
```

```
bash$ ./assert
Assertion failed: (false), function main, file assert.cc, line 4.
Abort trap: 6
```

```
Exception Type:  EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Application Specific Information:
Assertion failed: (false), function main, file assert.cc, line 4.
```

# What Is a Crash?

Google

Software: `abort()` vs. `__asm__("int3")`

```
int main(int argc, char* argv[]) {
  fprintf(stderr, "first\n");
  abort();
  fprintf(stderr, "second\n");
  return 0;
}
```

```
bash$ gdb -q abort3
(gdb) run
first
Program received signal SIGABRT, Aborted.
0x967983ba in __kill ()
(gdb) continue
Continuing.
Program terminated with signal SIGABRT, Aborted.
The program no longer exists.
(gdb)
```

# What Is a Crash?

Google

Software: `abort()` vs. `__asm__("int3")`

```
int main(int argc, char* argv[]) {
  fprintf(stderr, "first\n");
  __asm__("int3");
  fprintf(stderr, "second\n");
  return 0;
}
```

```
bash$ gdb -q int3
(gdb) run
first
Program received signal SIGTRAP, Trace/breakpoint trap.
main (argc=1, argv=0xbffffb2c) at int3.cc:4
4         fprintf(stderr, "second\n");
(gdb) continue
Continuing.
second
Program exited normally.
(gdb)
```

# Heap Corruption

Google

```
#include <stdlib.h>

int main(int argc, char* argv[]) {
  void* pointer = malloc(1024);
  free(pointer);
  free(pointer);
  return 0;
}
```

```
bash$ ./malloc_double_free
malloc_double_free(7693) malloc: *** error for object 0x79a94a00:
pointer being freed was not allocated
*** set a breakpoint in malloc_error_break to debug
Abort trap: 6

Exception Type:  EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Application Specific Information:
*** error for object 0x79a94a00: pointer being freed was not
allocated
```

# ObjCZombie

Google

chrome/common/mac/objc_zombie.mm (Chrome 24, paraphrased):

```
void ZombieObjectCrash(id object, SEL selector) {
  abort();
}

@interface CrZombie {
  Class isa;
}
@end

@implementation CrZombie
+ (void)initialize {
}

- (id)forwardingTargetForSelector:(SEL)selector {
  ZombieObjectCrash(self, selector);
  return nil;
}
@end
```

# ObjCZombie

Google

chrome/common/mac/objc_zombie.mm (Chrome 24, paraphrased):

```
const size_t zombieCount = 10000;
id zombieTreadmill[zombieCount];
size_t zombieIndex;
pthread_mutex_t zombieLock = PTHREAD_MUTEX_INITIALIZER;

void ZombieDealloc(id self, SEL _cmd) {
  objc_destructInstance(self);
#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wdeprecated-objc-isa-usage"
  self->isa = [CrZombie class];
#pragma clang diagnostic pop
  pthread_mutex_lock(&zombieLock);
  id zombieToFree = zombieTreadmill[zombieIndex];
  zombieTreadmill[zombieIndex] = self;
  zombieIndex = (zombieIndex + 1) % zombieCount;
  pthread_mutex_unlock(&zombieLock);
  if (zombieToFree)
    object_dispose(zombieToFree);
}
```

# ObjCZombie

Google

chrome/common/mac/objc_zombie.mm (Chrome 24, paraphrased):

```
void ZombieEnable() {
  Method m = class_getInstanceMethod([NSObject class],
                                     @selector(dealloc));
  if (!m)
    return;
  method_setImplementation(m, (IMP)ZombieDealloc);
}
```

# ObjCZombie

Google

# Out-of-Memory Killer

Google

```
void* SomeFunction() {
  const size_t size = 123456;
  char* new_string = (char*)malloc(size);
  assert(new_string);
  memset(new_string, '?', size - 1);
  new_string[size - 1] = '\0';

  char* copy_of_string = strdup(new_string);
  assert(copy_of_string);
}
```

# Out-of-Memory Killer

Google

base/process_util_mac.mm (Chrome 24, paraphrased):

```
typedef void* (*malloc_zone_malloc_t)(malloc_zone_t* zone,
                                      size_t size);
malloc_zone_malloc_t old_malloc;

void* oom_killer_malloc(malloc_zone_t* zone, size_t size) {
  void* result = old_malloc(zone, size);
  assert(result || !size);
  return result;
}

void EnableTerminationOnOutOfMemory() {
  malloc_zone_t* default_zone = malloc_default_zone();
  vm_address_t start = mach_vm_trunc_page(default_zone);
  vm_size_t length = (vm_address_t)default_zone - start +
                     sizeof(malloc_zone_t);
  old_malloc = default_zone->malloc;
  mprotect(start, length, PROT_READ | PROT_WRITE);
  default_zone->malloc = oom_killer_malloc;
  mprotect(start, length, PROT_READ);
}
```

# Out-of-Memory Killer

Google

base/process_util_mac.mm (Chrome 24, paraphrased):

```
CFAllocatorAllocateCallBack old_cfallocator_system_default;

void* oom_killer_cfallocator_system_default(CFIndex alloc_size,
                                            CFOptionFlags hint,
                                            void* info) {
  void* result = old_cfallocator_system_default(alloc_size,
                                                hint,
                                                info);
  assert(result);
  return result;
}

void EnableTerminationOnOutOfMemory() {
  CFAllocatorContext* context =
      ContextForCFAllocator(kCFAllocatorSystemDefault);
  old_cfallocator_system_default = context->allocate;
  context->allocate = oom_killer_cfallocator_system_default;
}
```

# Out-of-Memory Killer

Google

base/process_util_mac.mm (Chrome 24, paraphrased):

```
void oom_killer_new() {
  abort();
}

void EnableTerminationOnOutOfMemory() {
  std::set_new_handler(oom_killer_new);
}
```

# Out-of-Memory Killer

Google

base/process_util_mac.mm (Chrome 24, paraphrased):

```
typedef id (*allocWithZone_t)(id self, SEL _cmd, NSZone* zone);
allocWithZone_t old_allocWithZone;

id oom_killer_allocWithZone(id self, SEL _cmd, NSZone* zone) {
  id result = old_allocWithZone(self, _cmd, zone);
  assert(result);
  return result;
}

void EnableTerminationOnOutOfMemory() {
  Method method = class_getMethod([NSObject class],
                                  @selector(allocWithZone:);
  old_allocWithZone = method_getImplementation(method);
  method_setImplementation(method,
                           (IMP)oom_killer_allocWithZone);
}
```

# ObjCCast

Google

```
NSString* GetFilenameFromDictionary(NSDictionary* d) {
  return [d objectForKey:@"filename"];
}

int OpenFileForDictionary(NSString* root,
                          NSDictionary* d) {
  NSString* name = GetFilenameFromDictionary(d);
  NSString* path = [root stringByAppendingPathComponent:name];
  return open([path fileSystemRepresentation], O_RDONLY);
}
```

```
#0  0x95129947 in ___TERMINATING_DUE_TO_UNCAUGHT_EXCEPTION___ ()
#1  0x925f352e in objc_exception_throw ()
#2  0x9512cd9d in -[NSObject(NSObject) doesNotRecognizeSelector:]
()
#3  0x95075437 in ___forwarding___ ()
#4  0x950751e2 in __forwarding_prep_0___ ()
#5  0x983e2cca in -[NSString(NSPathUtilities)
stringByAppendingPathComponent:] ()
#6  0x00001d42 in OpenFileForDictionary (root=0x2020, d=0x16bb00)
at objccast.mm:27
```

# ObjCCast

Google

base/mac/foundation_util.h (Chrome 24, paraphrased):

```
template<typename T>
T* ObjCCast(id objc_val) {
  if ([objc_val isKindOfClass:[T class]])
    return reinterpret_cast<T*>(objc_val);
  return nil;
}

template<typename T>
T* ObjCCastStrict(id objc_val) {
  T* rv = ObjCCast<T>(objc_val);
  assert(!objc_val || rv);
  return rv;
}
```

# ObjCCast

Google

```
NSString* GetFilenameFromDictionary(NSDictionary* d) {
  return ObjCCastStrict<NSString>([d objectForKey:@"filename"]);
}

int OpenFileForDictionary(NSString* root,
                          NSDictionary* d) {
  NSString* name = GetFilenameFromDictionary(d);
  NSString* path = [root stringByAppendingPathComponent:name];
  return open([path fileSystemRepresentation], O_RDONLY);
}
```

```
#0  0x967983ba in __kill ()
#1  0x967974bc in kill$UNIX2003 ()
#2  0x92aea527 in abort ()
#3  0x92af8d37 in __assert_rtn ()
#4  0x00001d2f in ObjCCastStrict<NSString> (objc_val=0x177bb0) at
objccast.mm:14
#5  0x00001b29 in GetFilenameFromDictionary (d=0x177b10) at
objccast.mm:19
#6  0x00001b58 in OpenFileForDictionary (root=0x202c, d=0x177b10)
at objccast.mm:26
```

# CRSetCrashLogMessage

Google

```
struct CRAnnotations {
  int version;
  int unknown;
  const char* message;
};

CRAnnotations gCRAnnotations
    __attribute__((section("__DATA, __crash_info"))) = { 4 };

void CRSetCrashLogMessage(const char* message) {
  gCRAnnotations.message = message;
}

void AbortWithMessage(const char* message) {
  CRSetCrashLogMessage(message);
  fprintf(stderr, "%s\n", message);
  abort();
}
```

# CRSetCrashLogMessage

Google

```
int main(int argc, char* argv[]) {
  AbortWithMessage("kaboom");
  return 0;
}
```

```
bash$ ./crsetcrashlogmessage
kaboom
Abort trap: 6

Exception Type:  EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Application Specific Information:
kaboom
abort() called
```

# CRSetCrashLogMessage + ObjCCast

Google

```
template<typename T>
T* ObjCCastStrict(id objc_val) {
  T* rv = ObjCCast<T>(objc_val);
  if (objc_val && !rv) {
    char message[256];
    snprintf(message, sizeof(message),
             "expected %s, observed %s",
             [[T className] UTF8String],
             [[objc_val className] UTF8String]);
    AbortWithMessage(message);
  }
  return rv;
}
```

```
Exception Type:  EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Application Specific Information:
expected NSString, observed __NSCFNumber
abort() called
```