# Defending the Heap: Diagnosing Undefined Behavior in Dynamic Memory with *jkmalloc*

Jakob K. Kaivo

West Virginia University

2023-12-15

# Agenda

Motivation

Background

Research Goals

Approach

Results

Conclusion

# Motivation

# Motivation

- Undefined Behavior can result in **anything** happening
- Anything often means (relatively safe) **optimization**
- More and more frequently, it introduces **vulnerabilities**
- Vulnerabilities lead to **exploits**
- **Better tooling** can reduce or eliminate these exploits

# Background

# Undefined Behavior

- What is Undefined Behavior?
- Defined in ISO/IEC 9899:2018 (emphasis added):

  *behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which this document imposes* ***no requirements***

- More colorfully described on Usenet in 1992 (emphasis added):

  *Permissible undefined behavior ranges from ignoring the situation completely with unpredictable results, to* ***having demons fly out of your nose***

# Consequences of Undefined Behavior

- Optimizations
  - Dead code removal
  - Instruction reordering to avoid overflow

# Consequences of Undefined Behavior

- ~~Optimizations~~ Vulnerabilities
  - ~~Dead code removal~~ Removing `NULL` checks
  - ~~Instruction reordering to avoid overflow~~ Breaking boundary checks

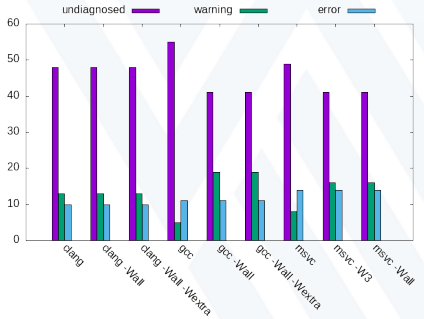
# Diagnosing Undefined Behavior

- Primarily the realm of **static analysis** tools
- Modern compilers now integrate some level of static analysis
  - **LLVM** has an explicit design goal of enabling static analysis
  - **GCC** has a legacy of very liberal compatibility, but is improving
- Some behavior is difficult or impossible to diagnose statically
- **Dynamic analysis** enables this
  - **Valgrind** is probably best known – runs programs in a heavily-instrumented virtual machine

# Undefined Behavior in the C Standard Library

- The C Standard lists **110 examples** of Undefined Behavior related to the Standard Library
- Many of these **require** dynamic analysis

# Compiler Diagnosis of Undefined Behavior in the Standard Library




Providing 72 sample programs that intentionally exhibit Undefined Behavior to modern compilers

# Dynamic Memory

- **Dynamic memory** (AKA heap) functions (i.e. `malloc()`, `free()`, etc.) account for seven entries in the list of Undefined Behavior
- Those entries are collectively responsible for a variety of **Heap Exploits**
  - Heap Overflow – #1 (write) and #6 (read) most dangerous CWEs of 2023
  - Heap Underflow
  - Use After Free – #4 most dangerous CWE of 2023
  - Double Free
- Existing efforts at diagnosing Undefined Behavior in the heap fall short
  - Static analysis is **unable** to identify behavior (e.g. compilers)
  - Too **heavy** (e.g. Valgrind, DieHard, DieHarder)
  - Diagnosis is **ambiguous** (e.g. Electric Fence, mimalloc, Scudo, snmalloc)

# Research Goals

# Challenges

- There are multiple previous efforts to enhance reliability of dynamic memory
- None have an emphasis on **unambiguously diagnosing** Undefined Behavior with **unmodified programs** while also being capable of integration with **production binaries**

| Characteristic | DieHard | DieHarder | Electric Fence | mimalloc | Scudo | snmalloc | Valgrind | ??????? |
|---|---|---|---|---|---|---|---|---|
| Unambiguous Diagnoses | - | - | - | - | - | - | ✓ | ? |
| Unmodified Binaries | some | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ? |
| Production Binary Integration | - | - | ✓ | ✓ | ✓ | ✓ | - | ? |

# Goals

- Provide **unambiguous diagnosis** of Undefined Behavior when encountered
- Require **minimal or no modification** to existing programs
  - Inject into **unmodified** programs using `LD_PRELOAD`
  - Additional diagnostic detail is available with **minimal** modification (i.e. one `#include`)
- Suitable for incorporation with **production binaries**
  - Enables detailed reporting from **non-technical** end-users (via copy-paste)

| Characteristic | DieHard | DieHarder | Electric Fence | mimalloc | Scudo | snmalloc | Valgrind | ***jkmalloc*** |
|---|---|---|---|---|---|---|---|---|
| Unambiguous Diagnoses | - | - | - | - | - | - | ✓ | ✓ |
| Unmodified Binaries | some | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Production Binary Integration | - | - | ✓ | ✓ | ✓ | ✓ | - | ✓ |

# Approach

# Traditional Approach

- Data and metadata are tightly packed
  - Overflow and underflow easily **corrupt metadata**
- Entire heap (including metadata) is **readable** and **writable**
  - There is **no protection** of the heap

# Use of Guard Pages

- Data is surrounded by inaccessible **guard pages**
  - Attempts to access results in **memory access violation** (segfault)
  - Prevents Overflow and Underflow
- Metadata is guarded, only accessible by heap management functions
  - Prevents **heap corruption**
- Page allocation from kernel provides **randomization** and a **sparse heap**
  - Prevents **Heap Spraying** attacks

# Testing

- Small test program that **intentionally exhibits** specific Undefined Behavior:
  - Zero Byte Allocation
  - Double Free
  - Invalid Free
  - Invalid Reallocation
  - Heap Overflow (controllable size)
  - Heap Underflow (controllable size)
  - Use After Free
- Verification using published heap-based **CVEs**

# Results

# Summary of Testing

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 Byte Allocation | - | D | -/A | - | - | - | - | - | F | - | D |
| Double Free | - | A | - | A | -/F | D | - | - | - | A | D |
| Invalid Free | - | A | F | A | -/A | D | F | F | D | A | D |
| Invalid Reallocation | - | D | F | D | -/A | D | F | F | D | D | D |
| Overflow (1 Byte) | - | F | -/A | - | - | - | - | - | - | - | D |
| Overflow (1 Page) | - | F | -/A | - | F/A | - | - | - | F | - | D |
| Underflow (1 Byte) | - | - | - | - | - | - | - | - | - | - | - |
| Underflow (1 Page) | - | F | -/F | A | -/A | - | - | F | - | - | D |
| Use After Free | -/O | F | O | - | -/O | O | - | - | F | O | D |

D = Unambiguous Diagnostic; A = Ambiguous Diagnostic; F = Segmentation Fault (No Additional Diagnostic); O = Freed Memory is Overwritten

# CVE-2021-3156: Heap-Based Buffer Overflow in Sudo

- Vulnerability known as Baron Samedit allows **unauthorized user** to escalate privileges to root
- Sudo with **no source modifications**, linked against *jkmalloc*, diagnoses the error rather than escalating privileges

```
$ ./sudoedit -s '\' `perl -e 'print "A" x 65536'`
Heap overflow detected: Segmentation fault (Invalid permissions for mapped object [0x7f65e3612000])
Allocation of size 65539 at 0x7f65e3601ffd, overflow at 0x7f65e3612000 (offset 65539)
Buffer begins with AAAA
```

# CVE-2022-43995: Heap-Based Buffer Overread in Sudo

- Vulnerability allows **unauthorized user** to read secret data from memory
- Sudo with **no source modifications**, linked against *jkmalloc*, diagnoses the error rather than escalating privileges

```
$ ./sudo id
Password:
Heap overflow detected: Segmentation fault (Invalid permissions for mapped object [0x7f48ac7bb003])
Allocation of size 5 at 0x7f48ac7baffb, overflow at 0x7f48ac7bb003 (offset 8)
Buffer begins with test
```

# Conclusion

# Contributions

- A new **dynamic memory manager**
  - Injectable into **unmodified** programs
  - Provides **unambiguous** diagnosis of Undefined Behavior
  - Provides **extended diagnosis** with **minimal modification** to exisiting sources
  - Can easily be incorporated into **production binaries**

# Future Work

- Additional **dynamic analysis-enabling** replacements for standard library functions
- Complementary **static analysis** tools for more thorough analysis of dynamic memory usage

# Questions?

# Zero Byte Allocation

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 Byte Allocation | - | D | -/A | - | - | - | - | - | F | - | D |

ElectricFence

```
ElectricFence Aborting: Allocating 0 bytes, probably a bug.
Illegal instruction
```

smimalloc

```
Aborted
```

Injected *jkmalloc*

```
Attempt to use 0-byte allocation: Segmentation fault (Invalid permissions for mapped object [0
    x7fa32773e000])
```

Explicit *jkmalloc*

```
Attempt to use 0-byte allocation: Segmentation fault (Invalid permissions for mapped object [0
    x7f1a4ecca000])
+++ main() (src/jktest.c:98)
```

# Double Free

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Double Free | - | A | - | A | -/F | D | - | - | - | A | D |

ElectricFence

```
ElectricFence Aborting: free(7f9ae14c8ff4): address not from malloc().
Illegal instruction
```

Scudo

```
Scudo ERROR: invalid chunk state when deallocating address 0x7f9a25a00630
Aborted
```

GNU libc

```
free(): double free detected in tcache 2
Aborted
```

macOS

```
jktest-dynamic(1449,0x1e4dc5ec0) malloc: *** error for object 0x60000194c030: pointer being freed was
      not allocated
jktest-dynamic(1449,0x1e4dc5ec0) malloc: *** set a breakpoint in malloc_error_break to debug
```

Injected *jkmalloc*

```
Double free() detected (0x7f1a3b7e6ff4)
```

Explicit *jkmalloc*

```
Double free() detected (0x7f600c9afff4)
+++ main() (src/jktest.c:98)
--- main() (src/jktest.c:114)
!!! main() (src/jktest.c:121)
```

# Invalid Free

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Invalid Free | - | A | F | A | -/A | D | F | F | D | A | D |

ElectricFence
```
ElectricFence Aborting: free(7ffd08550154): address not from malloc().
Illegal instruction
```

Scudo
```
Scudo ERROR: misaligned pointer when deallocating address 0x7fff4fcfc67c
Aborted
```

hardened snmalloc
```
Illegal instruction
```

GNU libc
```
munmap_chunk(): invalid pointer
Aborted
```

OpenBSD
```
jktest-dynamic(66802) in free(): bogus pointer (double free?) 0x7b17629e8d4c
Abort trap (core dumped)
```

macOS
```
jktest-dynamic(1456,0x1e4dc5ec0) malloc: *** error for object 0x16f40f638: pointer being freed was not
        allocated
jktest-dynamic(1456,0x1e4dc5ec0) malloc: *** set a breakpoint in malloc_error_break to debug
```

Injected *jkmalloc*
```
Attempt to free() non-dynamic address (0x7ffc9213f2c4)
```

Explicit *jkmalloc*
```
Attempt to free() non-dynamic address (0x7ffcc9fc9434)
!!! main() (src/jktest.c:126)
```

# Invalid Reallocation

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Invalid Reallocation | - | D | F | D | -/A | D | F | F | D | D | D |

ElectricFence
```
ElectricFence Aborting: realloc(7ffdd362b364, 12): address not from malloc().
Illegal instruction
```

Scudo
```
Scudo ERROR: misaligned pointer when reallocating address 0x7ffd45d5070c
Aborted
```

hardended snmalloc
```
Illegal instruction
```

GNU libc
```
realloc(): invalid pointer
Aborted
```

OpenBSD
```
jktest-dynamic(13014) in realloc(): bogus pointer (double free?) 0x720dd2bf04dc
Abort trap (core dumped)
```

macOS
```
jktest-dynamic(1467,0x1e4dc5ec0) malloc: *** error for object 0x16da63648: pointer being realloc'd was
        not allocated
jktest-dynamic(1467,0x1e4dc5ec0) malloc: *** set a breakpoint in malloc_error_break to debug
```

Injected *jkmalloc*
```
Attempt to realloc() non-dynamic address (0x7ffe587bff94)
```

Explicit *jkmalloc*
```
Attempt to realloc() non-dynamic address (0x7ffd875ccc44)
!!! main() (src/jktest.c:130)
```

# Heap Overflow (1 Byte)

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Overflow (1 Byte) | - | F | -/A | - | - | - | - | - | - | - | D |

smimalloc

```
Aborted
```

Injected *jkmalloc*

```
Heap overflow detected: Segmentation fault (Invalid permissions for mapped object [0x7f6402564000])
Allocation of size 12 at 0x7f6402563ff4, overflow at 0x7f6402564000 (offset 12)
Buffer begins with test string
```

Explicit *jkmalloc*

```
Heap overflow detected: Segmentation fault (Invalid permissions for mapped object [0x7f7fde8c4000])
Allocation of size 12 at 0x7f7fde8c3ff4, overflow at 0x7f7fde8c4000 (offset 12)
Buffer begins with test string
+++ main() (src/jktest.c:98)
```

# Heap Overflow (1 Page)

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Overflow (1 Page) | - | F | -/A | - | F/A | - | - | - | F | - | D |

smimalloc

```
Aborted
```

hardened snmalloc

```
Illegal instruction
```

Injected *jkmalloc*

```
Heap overflow detected: Segmentation fault (Invalid permissions for mapped object [0x7fc5e1480000])
Allocation of size 12 at 0x7fc5e147fff4, overflow at 0x7fc5e1480000 (offset 12)
Buffer begins with test string
```

Explicit *jkmalloc*

```
Heap overflow detected: Segmentation fault (Invalid permissions for mapped object [0x7f85a3731000])
Allocation of size 12 at 0x7f85a3730ff4, overflow at 0x7f85a3731000 (offset 12)
Buffer begins with test string
+++ main() (src/jktest.c:98)
```

# Heap Underflow (1 Byte)

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Underflow (1 Byte) | - | - | - | - | - | - | - | - | - | - | - |

- If the allocation is an exact multiple of the system page size, *jkmalloc* will diagnose 1 byte underflow
- Otherwise, the underflow is obscured by the necessary padding of the data page

# Heap Underflow (1 Page)

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Underflow (1 Page) | - | F | -/F | A | -/A | - | - | F | - | - | D |

Scudo

```
Scudo ERROR: corrupted chunk header at address 0x7f8442c00c10
Aborted
```

hardened snmalloc

```
Illegal instruction
```

Injected *jkmalloc*

```
Heap underflow detected: Segmentation fault (Invalid permissions for mapped object [0x7f094f4d2fff])
```

Explicit *jkmalloc*

```
Heap underflow detected: Segmentation fault (Invalid permissions for mapped object [0x7f152c390fff])
+++ main() (src/jktest.c:98)
```

# Use After Free

| Test Condition | DieHard/DieHarder | Electric Fence | mimalloc/smimalloc | Scudo | snmalloc/hardened | glibc | FreeBSD | NetBSD | OpenBSD | macOS | jkmalloc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Use After Free | -/O | F | O | - | -/O | O | - | - | F | O | D |

Injected *jkmalloc*

```
Use after free() detected: Segmentation fault (Invalid permissions for mapped object [0x7fc7e2cf7fe0])
```

Explicit *jkmalloc*

```
Use after free() detected: Segmentation fault (Invalid permissions for mapped object [0x7f87ece03fe0])
+++ main() (src/jktest.c:98)
—— main() (src/jktest.c:114)
```