Lab 7 Valgrind & Sanitizer

Environment

- Language: C++
- Debugger: gdb
- Compiler: g++ (GNU c++ compiler)
- g++ version: 9.4.0

Heap out-of-bounds

Source Code

```
int main(int argc, char **argv) {
    int SIZE = 50;
    int *arr = new int[SIZE];
    int res = arr[argc+SIZE]; // boom in read
    arr[argc+SIZE] = 1;  // boom in write

    delete [] arr;
    return res;
}
```

ASan report

```
==9174==ERROR: AddressSanitizer: heap-buffer-overflow on address
0x61100000010c at pc 0x5592fed452c4 bp 0x7ffe03bdf1a0 sp 0x7ffe03bdf190
READ of size 4 at 0x61100000010c thread T0
    #0 0x5592fed452c3 in main
/home/hscc_yuchen/Course/software_testing/lab6/heapOutOfBounds.cpp:4
    #1 0x7f3bb3c8f0b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)
    #2 0x5592fed4516d in _start
(/home/hscc_yuchen/Course/software_testing/lab6/a.out+0x116d)

0x61100000010c is located 4 bytes to the right of 200-byte region
[0x611000000040,0x611000000108)
allocated by thread T0 here:
    #0 0x7f3bb42b8787 in operator new[](unsigned long)
../../../../src/libsanitizer/asan/asan_new_delete.cc:107
    #1 0x5592fed45262 in main
/home/hscc_yuchen/Course/software_testing/lab6/heapOutOfBounds.cpp:3
    #2 0x7f3bb3c8f0b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)

SUMMARY: AddressSanitizer: heap-buffer-overflow
/home/hscc_yuchen/Course/software_testing/lab6/heapOutOfBounds.cpp:4 in
```

```
main
Shadow bytes around the buggy address:
  0x0c227fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff8000: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c227fff8010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c227fff8020: 00[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8060: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8070: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==9174==ABORTING
```

Valgrind report

```
# Command:
g++ -g heapOutOfBounds.cpp
valgrind ./a.out
```

```
==9281== Memcheck, a memory error detector
==9281== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9281== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==9281== Command: ./a.out
==9281==
==9281== Invalid read of size 4
==9281==    at 0x1091E0: main (heapOutOfBounds.cpp:4)
==9281==  Address 0x4db5d4c is 4 bytes after a block of size 200 alloc'd
```

```
==9281==    at 0x483C583: operator new[](unsigned long) (in
/usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==9281==    by 0x1091C2: main (heapOutOfBounds.cpp:3)
==9281==
==9281== Invalid write of size 4
==9281==    at 0x1091FE: main (heapOutOfBounds.cpp:5)
==9281==  Address 0x4db5d4c is 4 bytes after a block of size 200 alloc'd
==9281==    at 0x483C583: operator new[](unsigned long) (in
/usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==9281==    by 0x1091C2: main (heapOutOfBounds.cpp:3)
==9281==
==9281==
==9281== HEAP SUMMARY:
==9281==     in use at exit: 0 bytes in 0 blocks
==9281==   total heap usage: 2 allocs, 2 frees, 72,904 bytes allocated
==9281==
==9281== All heap blocks were freed -- no leaks are possible
==9281==
==9281== For lists of detected and suppressed errors, rerun with: -s
==9281== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

> **ASan能, Valgrind能**

Stack out-of-bounds

Source Code

```
int main(int argc, char** argv) {
    int stack_arr[50];

    stack_arr[50] = 1; // boom in write
    int res = stack_arr[argc+50]; // boom in read

    return res;
}
```

ASan report

```
==10417==ERROR: AddressSanitizer: stack-buffer-overflow on address
0x7ffe90ecfec8 at pc 0x557afd6b62d8 bp 0x7ffe90ecfda0 sp 0x7ffe90ecfd90
WRITE of size 4 at 0x7ffe90ecfec8 thread T0
    #0 0x557afd6b62d7 in main
/home/hscc_yuchen/Course/software_testing/lab6/stackOutOfBounds.cpp:4
    #1 0x7f2f5b67a0b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)
    #2 0x557afd6b612d in _start
(/home/hscc_yuchen/Course/software_testing/lab6/a.out+0x112d)
```

```
Address 0x7ffe90ecfec8 is located in stack of thread T0 at offset 248 in
frame
    #0 0x557afd6b61f8 in main
/home/hscc_yuchen/Course/software_testing/lab6/stackOutOfBounds.cpp:1

  This frame has 1 object(s):
    [48, 248) 'stack_arr' (line 2) <== Memory access at offset 248
overflows this variable
HINT: this may be a false positive if your program uses some custom stack
unwind mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow
/home/hscc_yuchen/Course/software_testing/lab6/stackOutOfBounds.cpp:4 in
main
Shadow bytes around the buggy address:
  0x1000521d1f80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000521d1f90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000521d1fa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000521d1fb0: 00 00 00 00 00 00 00 00 00 00 f1 f1 f1 f1 f1 f1
  0x1000521d1fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x1000521d1fd0: 00 00 00 00 00 00 00 00 00[f3]f3 f3 f3 f3 f3 f3
  0x1000521d1fe0: f3 f3 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000521d1ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000521d2000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000521d2010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000521d2020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==10417==ABORTING
```

Valgrind report

```
# Command:
g++ -g stackOutOfBounds.cpp
./a.out
```

```
==10556== Memcheck, a memory error detector
==10556== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10556== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright
info
==10556== Command: ./a.out
==10556==
*** stack smashing detected ***: terminated
==10556==
==10556== Process terminating with default action of signal 6 (SIGABRT)
==10556==    at 0x48A403B: raise (raise.c:51)
==10556==    by 0x4883858: abort (abort.c:79)
==10556==    by 0x48EE29D: __libc_message (libc_fatal.c:155)
==10556==    by 0x4990AE9: __fortify_fail (fortify_fail.c:26)
==10556==    by 0x4990AB5: __stack_chk_fail (stack_chk_fail.c:24)
==10556==    by 0x1091AC: main (stackOutOfBounds.cpp:8)
==10556==
==10556== HEAP SUMMARY:
==10556==     in use at exit: 0 bytes in 0 blocks
==10556==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==10556==
==10556== All heap blocks were freed -- no leaks are possible
==10556==
==10556== For lists of detected and suppressed errors, rerun with: -s
==10556== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Aborted (core dumped)
```

> ASan能, Valgrind不能

Global out-of-bounds

Source Code

```
int global_arr[50] = {0};

int main(int argv, char** argc) {
    global_arr[50] = 1; // boom in write
    int res = global_arr[50]; // boom in read

    return res;
}
```

ASan report

```
==10936==ERROR: AddressSanitizer: global-buffer-overflow on address
0x564e243c3168 at pc 0x564e243c020e bp 0x7ffcb5c77f50 sp 0x7ffcb5c77f40
WRITE of size 4 at 0x564e243c3168 thread T0
    #0 0x564e243c020d in main
/home/hscc_yuchen/Course/software_testing/lab6/globalOutOfBounds.cpp:4
    #1 0x7fb15a18a0b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)
    #2 0x564e243c010d in _start
(/home/hscc_yuchen/Course/software_testing/lab6/a.out+0x110d)

0x564e243c3168 is located 0 bytes to the right of global variable
'global_arr' defined in 'globalOutOfBounds.cpp:1:5' (0x564e243c30a0) of
size 200
SUMMARY: AddressSanitizer: global-buffer-overflow
/home/hscc_yuchen/Course/software_testing/lab6/globalOutOfBounds.cpp:4 in
main
Shadow bytes around the buggy address:
  0x0aca448705d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0aca448705e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0aca448705f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0aca44870600: 00 00 00 00 00 00 00 00 f9 f9 f9 f9 f9 f9 f9 f9
  0x0aca44870610: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0aca44870620: 00 00 00 00 00 00 00 00 00 00 00 00 00[f9]f9 f9
  0x0aca44870630: f9 f9 f9 f9 00 00 00 00 00 00 00 00 00 00 00 00
  0x0aca44870640: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0aca44870650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0aca44870660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0aca44870670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==10936==ABORTING
```

Valgrind report

```
==10980== Memcheck, a memory error detector
==10980== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10980== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright
info
==10980== Command: ./a.out
==10980==
==10980==
==10980== HEAP SUMMARY:
==10980==     in use at exit: 0 bytes in 0 blocks
==10980==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==10980==
==10980== All heap blocks were freed -- no leaks are possible
==10980==
==10980== For lists of detected and suppressed errors, rerun with: -s
==10980== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

> ASan能, Valgrind不能

Use after free

Source Code

```
int main(int argc, char** argv) {
    int *arr = new int[50];
    delete [] arr;

    int res = arr[1]; // boom
    return res;
}
```

ASan report

```
==11174==ERROR: AddressSanitizer: heap-use-after-free on address
0x611000000044 at pc 0x55d2b2d15238 bp 0x7ffdf28fc460 sp 0x7ffdf28fc450
READ of size 4 at 0x611000000044 thread T0
    #0 0x55d2b2d15237 in main
/home/hscc_yuchen/Course/software_testing/lab6/useAfterFree.cpp:5
    #1 0x7fa0b9f7b0b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)
    #2 0x55d2b2d1510d in _start
(/home/hscc_yuchen/Course/software_testing/lab6/a.out+0x110d)

0x611000000044 is located 4 bytes inside of 200-byte region
[0x611000000040,0x611000000108)
freed by thread T0 here:
    #0 0x7fa0ba5a56ef in operator delete[](void*)
../../../../src/libsanitizer/asan/asan_new_delete.cc:168
```

```
    #1 0x55d2b2d151fc in main
/home/hscc_yuchen/Course/software_testing/lab6/useAfterFree.cpp:3
    #2 0x7fa0b9f7b0b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)

previously allocated by thread T0 here:
    #0 0x7fa0ba5a4787 in operator new[](unsigned long)
../../../../src/libsanitizer/asan/asan_new_delete.cc:107
    #1 0x55d2b2d151e5 in main
/home/hscc_yuchen/Course/software_testing/lab6/useAfterFree.cpp:2
    #2 0x7fa0b9f7b0b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)

SUMMARY: AddressSanitizer: heap-use-after-free
/home/hscc_yuchen/Course/software_testing/lab6/useAfterFree.cpp:5 in main
Shadow bytes around the buggy address:
  0x0c227fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c227fff8000: fa fa fa fa fa fa fa fa[fd]fd fd fd fd fd fd fd
  0x0c227fff8010: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c227fff8020: fd fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==11174==ABORTING
```

Valgrind report

```
==11203== Memcheck, a memory error detector
==11203== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==11203== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright
info
==11203== Command: ./a.out
==11203==
==11203== Invalid read of size 4
==11203==    at 0x1091A1: main (useAfterFree.cpp:5)
==11203==  Address 0x4db5c84 is 4 bytes inside a block of size 200 free'd
==11203==    at 0x483D74F: operator delete[](void*) (in /usr/lib/x86_64-
linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==11203==    by 0x10919C: main (useAfterFree.cpp:3)
==11203==  Block was alloc'd at
==11203==    at 0x483C583: operator new[](unsigned long) (in
/usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==11203==    by 0x109185: main (useAfterFree.cpp:2)
==11203==
==11203==
==11203== HEAP SUMMARY:
==11203==     in use at exit: 0 bytes in 0 blocks
==11203==   total heap usage: 2 allocs, 2 frees, 72,904 bytes allocated
==11203==
==11203== All heap blocks were freed -- no leaks are possible
==11203==
==11203== For lists of detected and suppressed errors, rerun with: -s
==11203== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

> ASan能, Valgrind能

Use after return

Source Code

```
char* x;

void foo() {
    char arr[50];
    x = &arr[13];
}

int main() {

    foo();
    *x = 'c'; // boom

    return 0;
}
```

ASan report

```
# Command:
g++ -g -fsanitize=address useAfterReturn.cpp
ASAN_OPTIONS=detect_stack_use_after_return=1 ./a.out
```

```
==12615==ERROR: AddressSanitizer: stack-use-after-return on address
0x7f5052b0802d at pc 0x564f2313d35f bp 0x7ffda9d31130 sp 0x7ffda9d31120
WRITE of size 1 at 0x7f5052b0802d thread T0
    #0 0x564f2313d35e in main
/home/hscc_yuchen/Course/software_testing/lab6/useAfterReturn.cpp:11
    #1 0x7f50560c50b2 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x240b2)
    #2 0x564f2313d14d in _start
(/home/hscc_yuchen/Course/software_testing/lab6/a.out+0x114d)

Address 0x7f5052b0802d is located in stack of thread T0 at offset 45 in
frame
    #0 0x564f2313d218 in foo()
/home/hscc_yuchen/Course/software_testing/lab6/useAfterReturn.cpp:3

  This frame has 1 object(s):
    [32, 82) 'arr' (line 4) <== Memory access at offset 45 is inside this
variable
HINT: this may be a false positive if your program uses some custom stack
unwind mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-use-after-return
/home/hscc_yuchen/Course/software_testing/lab6/useAfterReturn.cpp:11 in
main
Shadow bytes around the buggy address:
  0x0fea8a558fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a558fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a558fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a558fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a558ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0fea8a559000: f5 f5 f5 f5 f5[f5]f5 f5 f5 f5 f5 f5 f5 f5 f5 f5
  0x0fea8a559010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a559020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a559030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a559040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fea8a559050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
```

```
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==12615==ABORTING
```

Valgrind report

```
# Command:
g++ -g useAfterReturn.cpp
valgrind ./a.out
```

```
==12509== Memcheck, a memory error detector
==12509== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12509== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright
info
==12509== Command: ./a.out
==12509==
==12509==
==12509== HEAP SUMMARY:
==12509==     in use at exit: 0 bytes in 0 blocks
==12509==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==12509==
==12509== All heap blocks were freed -- no leaks are possible
==12509==
==12509== For lists of detected and suppressed errors, rerun with: -s
==12509== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

> ASan能, Valgrind不能

越過redzone做讀寫

Source Code

```
int main(int argc, char** argv) {
    char *a = new char[8];
    char *b = new char[8];
```

```
    a[35] = 'c';

    delete [] a;
    delete [] b;
    return 0;
}
```

ASan report

```
無
```

> 越過redzone後做讀寫，ASan無法偵測到