Lab7 Valgrind & Sanitizer

Property	
Tags	C++ DEBUG Sanitizer Valgrind

Environments

使用clang進行測試

```
clang --version
- clang version 10.0.0-4ubuntu1
- Target: x86_64-pc-linux-gnu
```

Lab7 - Requirement 1

Heap out-of-bounds read/write

Source code:

```
int main(int argc, char **argv) {
  int *array = new int[100];
  array[0] = 0;
  int res = array[argc + 100];  // BOOM
  delete [] array;
  return res;
}
```

Linux command (Address Sanitizer, ASan):

```
# Defualt output filename: a.out
clang++ -g -fsanitize=address HeapOutOfBound.cpp
# run
./a.out

# Specify output filename
clang++ -g -fsanitize=address HeapOutOfBound.cpp -o {out_filename}.{extension}
# run
./{out_filename}.{extension}
```

ASan result:

```
// ASan report
==818517==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6140000001d4 at pc 0x
0000004c6b3b bp 0x7fffe215d850 sp 0x7fffe215d848
READ of size 4 at 0x6140000001d4 thread T0
    #0 0x4c6b3a in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/HeapOutOfBound.cpp:4:13
    #1 0x7f65f7e070b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #2 0x41c2dd in _start (/media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Test
ing-NYCU-2021/Lab7/a.out+0x41c2dd)

0x6140000001d4 is located 4 bytes to the right of 400-byte region [0x614000000040,0x614000
0001d0)
allocated by thread T0 here:
    #0 0x4c429d in operator new[](unsigned long) (/media/yung/18194E037A622565/Yung/Softwa
reTesting/Software-Testing-NYCU-2021/Lab7/a.out+0x4c429d)
    #1 0x4c6a8f in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/HeapOutOfBound.cpp:2:16
    #2 0x7f65f7e070b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16

SUMMARY: AddressSanitizer: heap-buffer-overflow /media/yung/18194E037A622565/Yung/Software
Testing/Software-Testing-NYCU-2021/Lab7/HeapOutOfBound.cpp:4:13 in main
Shadow bytes around the buggy address:
  0x0c287fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff8000: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c287fff8010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff8020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c287fff8030: 00 00 00 00 00 00 00 00 00 00[fa]fa fa fa fa fa
  0x0c287fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff8060: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff8070: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff8080: 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
==818517==ABORTING
```

Linux command (Valgrind):

```
clang++ -g HeapOutOfBound.cpp
# run
valgrind ./a.out
```

Valgrind result:

```
// Valgrind report
==818367== Memcheck, a memory error detector
==818367== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==818367== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==818367== Command: ./a.out
==818367==
==818367== Invalid read of size 4
==818367==    at 0x40117B: main (HeapOutOfBound.cpp:4)
==818367==  Address 0x4db1e14 is 4 bytes after a block of size 400 alloc'd
==818367==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/va
lgrind/vgpreload_memcheck-amd64-linux.so)
==818367==    by 0x40115F: main (HeapOutOfBound.cpp:2)
==818367==
==818367==
==818367== HEAP SUMMARY:
==818367==     in use at exit: 0 bytes in 0 blocks
==818367==   total heap usage: 2 allocs, 2 frees, 73,104 bytes allocated
==818367==
==818367== All heap blocks were freed -- no leaks are possible
==818367==
==818367== For lists of detected and suppressed errors, rerun with: -s
==818367== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

> **ASan 能 , Valgrind 能**

Stack out-of-bounds read/write

```
// StackOutOfBounds.cpp
int main(int argc, char **argv) {
  int stack_array[100];
  stack_array[1] = 0;
  return stack_array[argc + 100];  // BOOM
}
```

```
// ASan reprot
==849583==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffd6da5d7c4 at pc 0
x0000004c6c19 bp 0x7ffd6da5d750 sp 0x7ffd6da5d748
READ of size 4 at 0x7ffd6da5d7c4 thread T0
    #0 0x4c6c18 in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/Lab7_2.cpp:5:13
    #1 0x7f7f9eeee0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #2 0x41c2dd in _start (/media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Test
ing-NYCU-2021/Lab7/a.out+0x41c2dd)

Address 0x7ffd6da5d7c4 is located in stack of thread T0 at offset 100 in frame
    #0 0x4c6a7f in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/Lab7_2.cpp:1

  This frame has 1 object(s):
    [32, 96) 'a' (line 2) <== Memory access at offset 100 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanis
m, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /media/yung/18194E037A622565/Yung/Softwar
eTesting/Software-Testing-NYCU-2021/Lab7/Lab7_2.cpp:5:13 in main
Shadow bytes around the buggy address:
  0x10002db43aa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43ab0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43ac0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43ad0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43ae0: 00 00 00 00 00 00 00 00 00 00 00 00 f1 f1 f1 f1
=>0x10002db43af0: 00 00 00 00 00 00 00 00[f3]f3 f3 f3 00 00 00 00
  0x10002db43b00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43b10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43b20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43b30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10002db43b40: 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
==849583==ABORTING
```

```
// Valgrind report
==850337== Memcheck, a memory error detector
==850337== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==850337== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==850337== Command: ./a.out
==850337==
==850337==
==850337== HEAP SUMMARY:
==850337==     in use at exit: 0 bytes in 0 blocks
==850337==   total heap usage: 1 allocs, 1 frees, 72,704 bytes allocated
==850337==
==850337== All heap blocks were freed -- no leaks are possible
==850337==
==850337== For lists of detected and suppressed errors, rerun with: -s
==850337== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

> ASan能, Valgrind不能

為何Valgrind不能，可參考這裡:
https://stackoverflow.com/questions/29842747/valgrind-wont-detect-buffer-overflow

---

Global out-of-bounds read/write

```
// GlobalOutOfBounds.cpp
int global_array[100] = {-1};
int main(int argc, char **argv) {
  return global_array[argc + 100];  // BOOM
}
```

```
// ASan report
==851839==ERROR: AddressSanitizer: global-buffer-overflow on address 0x0000004fad34 at pc
 0x0000004c6ae4 bp 0x7ffc55b635a0 sp 0x7ffc55b63598
READ of size 4 at 0x0000004fad34 thread T0
    #0 0x4c6ae3 in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/GlobalOutOfBounds.cpp:3:10
    #1 0x7f4ca0c1d0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #2 0x41c2dd in _start (/media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Test
ing-NYCU-2021/Lab7/a.out+0x41c2dd)

0x0000004fad34 is located 4 bytes to the right of global variable 'global_array' defined i
n 'GlobalOutOfBounds.cpp:1:5' (0x4faba0) of size 400
SUMMARY: AddressSanitizer: global-buffer-overflow /media/yung/18194E037A622565/Yung/Softwa
reTesting/Software-Testing-NYCU-2021/Lab7/GlobalOutOfBounds.cpp:3:10 in main
Shadow bytes around the buggy address:
  0x000080097550: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x000080097560: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x000080097570: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x000080097580: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x000080097590: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0000800975a0: 00 00 00 00 00 00[f9]f9 f9 f9 f9 f9 f9 f9 f9 f9
  0x0000800975b0: f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 00 00 00 00
  0x0000800975c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800975d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800975e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800975f0: 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
==851839==ABORTING
```

```
// Valgrind report
==852215== Memcheck, a memory error detector
==852215== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==852215== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==852215== Command: ./a.out
==852215==
==852215==
==852215== HEAP SUMMARY:
==852215==     in use at exit: 0 bytes in 0 blocks
==852215==   total heap usage: 1 allocs, 1 frees, 72,704 bytes allocated
==852215==
==852215== All heap blocks were freed -- no leaks are possible
==852215==
==852215== For lists of detected and suppressed errors, rerun with: -s
==852215== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

> ASan能, Valgrind不能

Use-after-free

```
// Use-after-free.cpp
int main(int argc, char **argv) {
  int *array = new int[100];
  delete [] array;
  return array[argc];  // BOOM
}
```

```
// ASan report
==853530==ERROR: AddressSanitizer: heap-use-after-free on address 0x614000000044 at pc 0x0
000004c6b02 bp 0x7ffcdcfd4c90 sp 0x7ffcdcfd4c88
READ of size 4 at 0x614000000044 thread T0
    #0 0x4c6b01 in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/Use-after-free.cpp:4:10
    #1 0x7fd21051c0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #2 0x41c2dd in _start (/media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Test
ing-NYCU-2021/Lab7/a.out+0x41c2dd)

0x614000000044 is located 4 bytes inside of 400-byte region [0x614000000040,0x6140000001d
0)
freed by thread T0 here:
    #0 0x4c4aed in operator delete[](void*) (/media/yung/18194E037A622565/Yung/SoftwareTes
ting/Software-Testing-NYCU-2021/Lab7/a.out+0x4c4aed)
    #1 0x4c6ab1 in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
```

```
-NYCU-2021/Lab7/Use-after-free.cpp:3:3
    #2 0x7fd21051c0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16

previously allocated by thread T0 here:
    #0 0x4c429d in operator new[](unsigned long) (/media/yung/18194E037A622565/Yung/Softwa
reTesting/Software-Testing-NYCU-2021/Lab7/a.out+0x4c429d)
    #1 0x4c6a8f in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/Use-after-free.cpp:2:16
    #2 0x7fd21051c0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16

SUMMARY: AddressSanitizer: heap-use-after-free /media/yung/18194E037A622565/Yung/SoftwareT
esting/Software-Testing-NYCU-2021/Lab7/Use-after-free.cpp:4:10 in main
Shadow bytes around the buggy address:
  0x0c287fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c287fff8000: fa fa fa fa fa fa fa fa[fd]fd fd fd fd fd fd fd
  0x0c287fff8010: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c287fff8020: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c287fff8030: fd fd fd fd fd fd fd fd fd fd fa fa fa fa fa fa
  0x0c287fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff8050: 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
==853530==ABORTING
```

```
// Valgrind report
==853913== Memcheck, a memory error detector
==853913== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
```

```
==853913== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==853913== Command: ./a.out
==853913==
==853913== Invalid read of size 4
==853913==    at 0x40118A: main (Use-after-free.cpp:4)
==853913==  Address 0x4db1c84 is 4 bytes inside a block of size 400 free'd
==853913==    at 0x483D74F: operator delete[](void*) (in /usr/lib/x86_64-linux-gnu/valgrin
d/vgpreload_memcheck-amd64-linux.so)
==853913==    by 0x401181: main (Use-after-free.cpp:3)
==853913==  Block was alloc'd at
==853913==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/va
lgrind/vgpreload_memcheck-amd64-linux.so)
==853913==    by 0x40115F: main (Use-after-free.cpp:2)
==853913==
==853913==
==853913== HEAP SUMMARY:
==853913==     in use at exit: 0 bytes in 0 blocks
==853913==   total heap usage: 2 allocs, 2 frees, 73,104 bytes allocated
==853913==
==853913== All heap blocks were freed -- no leaks are possible
==853913==
==853913== For lists of detected and suppressed errors, rerun with: -s
==853913== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

> **ASan 能 , Valgrind 能**

Use-after-return

```
// UseAfterReturn.cpp
// By default, AddressSanitizer does not try to detect
// stack-use-after-return bugs.
// It may still find such bugs occasionally
// and report them as a hard-to-explain stack-buffer-overflow.

// You need to run the test with ASAN_OPTIONS=detect_stack_use_after_return=1

// AddressSanitizer currently does not attempt to detect these bugs by default, only with
 an additional flag run-time: ASAN_OPTIONS=detect_stack_use_after_return=1

int *ptr;
__attribute__((noinline))
void FunctionThatEscapesLocalObject() {
  int local[100];
  ptr = &local[0];
}

int main(int argc, char **argv) {
```

```
  FunctionThatEscapesLocalObject();
  return ptr[argc];
}
```

💡 By default, AddressSanitizer does not try to detect stack-use-after-return bugs. 在Run-time時要下 ASAN_OPTIONS=detect_stack_use_after_return=1

```
# clang with Sanitizer
clang++ -g -fsanitize=address UseAfterReturn.cpp
# run
ASAN_OPTIONS=detect_stack_use_after_return=1 ./a.out
```

```
// ASan report
==855797==ERROR: AddressSanitizer: stack-use-after-return on address 0x7f2b10845024 at pc
 0x0000004c6ccf bp 0x7ffc5d3ac050 sp 0x7ffc5d3ac048
READ of size 4 at 0x7f2b10845024 thread T0
    #0 0x4c6cce in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/UseAfterReturn.cpp:17:10
    #1 0x7f2b13bcf0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #2 0x41c2dd in _start (/media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Test
ing-NYCU-2021/Lab7/a.out+0x41c2dd)

Address 0x7f2b10845024 is located in stack of thread T0 at offset 36 in frame
    #0 0x4c6a7f in FunctionThatEscapesLocalObject() /media/yung/18194E037A622565/Yung/Soft
wareTesting/Software-Testing-NYCU-2021/Lab7/UseAfterReturn.cpp:10

  This frame has 1 object(s):
    [32, 432) 'local' (line 11) <== Memory access at offset 36 is inside this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanis
m, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-use-after-return /media/yung/18194E037A622565/Yung/Softwa
reTesting/Software-Testing-NYCU-2021/Lab7/UseAfterReturn.cpp:17:10 in main
Shadow bytes around the buggy address:
  0x0fe5e21009b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fe5e21009c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fe5e21009d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fe5e21009e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fe5e21009f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0fe5e2100a00: f5 f5 f5 f5[f5]f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5
  0x0fe5e2100a10: f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5
  0x0fe5e2100a20: f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5
  0x0fe5e2100a30: f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5 f5
```

```
  0x0fe5e2100a40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0fe5e2100a50: 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
==855797==ABORTING
```

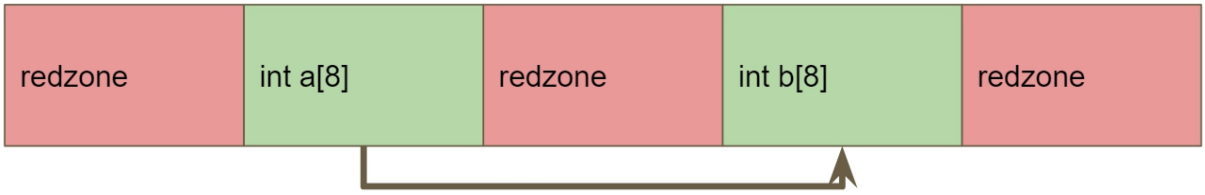
```
// Valgrind report
==857273== Memcheck, a memory error detector
==857273== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==857273== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==857273== Command: ./a.out
==857273==
==857273== Invalid read of size 4
==857273==    at 0x401167: main (UseAfterReturn.cpp:17)
==857273==  Address 0x1ffefffd84 is on thread 1's stack
==857273==  412 bytes below stack pointer
==857273==
==857273==
==857273== HEAP SUMMARY:
==857273==     in use at exit: 0 bytes in 0 blocks
==857273==   total heap usage: 1 allocs, 1 frees, 72,704 bytes allocated
==857273==
==857273== All heap blocks were freed -- no leaks are possible
==857273==
==857273== For lists of detected and suppressed errors, rerun with: -s
==857273== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

ASan 能 , Valgrind 能

Lab7 - Requirement2

Background:

檢查下列情況，若a array 越界存取 b array，ASan是否可檢測出。



> 結論是ASan找不到，使用兩個Test cases來驗證這個結論

找的到錯誤的情況:

```
// Lab7_2.cpp
int main(int argc, char **argv) {
  int *a = new int[8];
  int *b = new int[8];
  int res = a[argc + 8];  // BOOM
  return res;
}
```

```
// ASan report
==845678==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x603000000034 at pc 0x
0000004c6afa bp 0x7ffda3d112f0 sp 0x7ffda3d112e8
READ of size 4 at 0x603000000034 thread T0
    #0 0x4c6af9 in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/Lab7_2.cpp:4:13
    #1 0x7f9ea8d7e0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #2 0x41c2dd in _start (/media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Test
ing-NYCU-2021/Lab7/a.out+0x41c2dd)

0x603000000034 is located 4 bytes to the right of 32-byte region [0x603000000010,0x6030000
00030)
allocated by thread T0 here:
    #0 0x4c429d in operator new[](unsigned long) (/media/yung/18194E037A622565/Yung/Softwa
reTesting/Software-Testing-NYCU-2021/Lab7/a.out+0x4c429d)
    #1 0x4c6a8f in main /media/yung/18194E037A622565/Yung/SoftwareTesting/Software-Testing
-NYCU-2021/Lab7/Lab7_2.cpp:2:12
```

```
    #2 0x7f9ea8d7e0b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16

SUMMARY: AddressSanitizer: heap-buffer-overflow /media/yung/18194E037A622565/Yung/Software
Testing/Software-Testing-NYCU-2021/Lab7/Lab7_2.cpp:4:13 in main
Shadow bytes around the buggy address:
  0x0c067fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c067fff8000: fa fa 00 00 00 00[fa]fa 00 00 00 00 fa fa fa fa
  0x0c067fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8050: 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
==845678==ABORTING
```

Asan找不到錯誤的情況:

```
// Lab7_2.cpp
int main(int argc, char **argv) {
  int *a = new int[8];
  int *b = new int[8];
  int res = a[argc + 12];  // Safe
  delete [] a;
  delete [] b;
```

```
  return res;
}
```

```
// No ASan report
```