Lab_6

環境

```
ubuntu:20.04
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
valgrind-3.15.0
```

Heap out-of-bounds

code

```c
#include <stdlib.h>
#include <stdio.h>

int main(){
    char *myHeap = malloc(4);
    printf("%c\n", myHeap[4]);
    free(myHeap);

    return 0;
}

// gcc HeapOutOfBounds.c -o heap
// gcc -fsanitize=address HeapOutOfBounds.c -o heap_ASan

// ./heap_ASan
// valgrind ./heap
```

ASan report

```
=================================================================
==102==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000000014 at pc
0x55bcac68227f bp 0x7ffee0cffc80 sp 0x7ffee0cffc70
READ of size 1 at 0x602000000014 thread T0
    #0 0x55bcac68227e in main (/home/lab6/share/heap_ASan+0x127e)
    #1 0x7f0c819b20b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)
    #2 0x55bcac68216d in _start (/home/lab6/share/heap_ASan+0x116d)

0x602000000014 is located 0 bytes to the right of 4-byte region
[0x602000000010,0x602000000014)
allocated by thread T0 here:
    #0 0x7f0c81c8d808 in __interceptor_malloc
../../../../src/libsanitizer/asan/asan_malloc_linux.cc:144
    #1 0x55bcac68223e in main (/home/lab6/share/heap_ASan+0x123e)
    #2 0x7f0c819b20b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)

SUMMARY: AddressSanitizer: heap-buffer-overflow (/home/lab6/share/heap_ASan+0x127e)
```

```
in main
Shadow bytes around the buggy address:
  0x0c047fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c047fff8000: fa fa[04]fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8050: 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
==102==ABORTING
```

valgrind report

```
==104== Memcheck, a memory error detector
==104== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==104== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==104== Command: ./heap
==104==
==104== Invalid read of size 1
==104==    at 0x1091AB: main (in /home/lab6/share/heap)
==104==  Address 0x4a42044 is 0 bytes after a block of size 4 alloc'd
==104==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-
gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==104==    by 0x10919E: main (in /home/lab6/share/heap)
==104==

==104==
```

```
==104== HEAP SUMMARY:
==104==     in use at exit: 0 bytes in 0 blocks
==104==   total heap usage: 2 allocs, 2 frees, 1,028 bytes allocated
==104==
==104== All heap blocks were freed -- no leaks are possible
==104==
==104== For lists of detected and suppressed errors, rerun with: -s
==104== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

ASan 能, valgrind 能

Stack out-of-bounds

code

```c
#include <stdlib.h>
#include <stdio.h>

int main(){
    int myStack[3] = {1, 2, 3};
    printf("%d\n", myStack[3]);

    return 0;
}

// gcc StackOutOfBounds.c -o stack
// gcc -fsanitize=address StackOutOfBounds.c -o stack_ASan

// ./stack_ASan
// valgrind ./stack
```

ASan report

```
=================================================================
==46==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7fffb4ae8bcc at pc
0x55e0032c43c0 bp 0x7fffb4ae8b90 sp 0x7fffb4ae8b80
READ of size 4 at 0x7fffb4ae8bcc thread T0
    #0 0x55e0032c43bf in main (/home/lab6/share/stack_ASan+0x13bf)
    #1 0x7f705ceb40b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)
    #2 0x55e0032c418d in _start (/home/lab6/share/stack_ASan+0x118d)

Address 0x7fffb4ae8bcc is located in stack of thread T0 at offset 44 in frame
    #0 0x55e0032c4258 in main (/home/lab6/share/stack_ASan+0x1258)

  This frame has 1 object(s):
    [32, 44) 'myStack' (line 5) <== Memory access at offset 44 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/lab6/share/stack_ASan+0x13bf) in main
Shadow bytes around the buggy address:
  0x100076955120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100076955130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100076955140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100076955150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100076955160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100076955170: 00 00 00 00 f1 f1 f1 f1 00[04]f3 f3 00 00 00 00
  0x100076955180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100076955190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000769551a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000769551b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000769551c0: 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
==46==ABORTING
```

valgrind report

```
==48== Memcheck, a memory error detector
==48== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==48== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==48== Command: ./stack
==48==
549488128
==48==
==48== HEAP SUMMARY:
==48==     in use at exit: 0 bytes in 0 blocks
==48==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==48==
==48== All heap blocks were freed -- no leaks are possible
```

```
==48==
==48== For lists of detected and suppressed errors, rerun with: -s
==48== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

ASan 能, valgrind 不能

Global out-of-bounds

```c
#include <stdlib.h>
#include <stdio.h>

int myGlobal[3] = {1, 2, 3};

int main(){
    printf("%d\n", myGlobal[3]);

    return 0;
}

// gcc GlobalOutOfBounds.c -o global
// gcc -fsanitize=address GlobalOutOfBounds.c -o global_ASan

// ./global_ASan
// valgrind ./global
```

ASan report

```
=================================================================
==61==ERROR: AddressSanitizer: global-buffer-overflow on address 0x55c5235fd02c at
pc 0x55c5235fa22b bp 0x7ffffc3bc0e0 sp 0x7ffffc3bc0d0
READ of size 4 at 0x55c5235fd02c thread T0
    #0 0x55c5235fa22a in main (/home/lab6/share/global_ASan+0x122a)
    #1 0x7f6cfa3600b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)
    #2 0x55c5235fa12d in _start (/home/lab6/share/global_ASan+0x112d)

0x55c5235fd02c is located 0 bytes to the right of global variable 'myGlobal' defined
in 'GlobalOutOfBounds.c:4:5' (0x55c5235fd020) of size 12
SUMMARY: AddressSanitizer: global-buffer-overflow
(/home/lab6/share/global_ASan+0x122a) in main
Shadow bytes around the buggy address:
  0x0ab9246b79b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab9246b79c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab9246b79d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab9246b79e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab9246b79f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0ab9246b7a00: 00 00 00 00 00[04]f9 f9 f9 f9 f9 f9 00 00 00 00
  0x0ab9246b7a10: f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9 f9
  0x0ab9246b7a20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
  0x0ab9246b7a30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab9246b7a40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab9246b7a50: 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
==61==ABORTING
```

valgrind report

```
==62== Memcheck, a memory error detector
==62== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==62== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==62== Command: ./global
==62==
0
==62==
==62== HEAP SUMMARY:
==62==     in use at exit: 0 bytes in 0 blocks
==62==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==62==
==62== All heap blocks were freed -- no leaks are possible
==62==
==62== For lists of detected and suppressed errors, rerun with: -s
==62== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

ASan 能, valgrind 不能

Use-after-free

code

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int *myHeap = malloc(2);
    free(myHeap);
    int used_after_free = myHeap[0];
    return 0;
}


// gcc UseAfterFree.c -o free
// gcc -fsanitize=address UseAfterFree.c -o free_ASan

// ./free_ASan
// valgrind ./free
```

ASan report

```
=================================================================
==688==ERROR: AddressSanitizer: heap-use-after-free on address 0x602000000010 at pc
0x560f0c29f226 bp 0x7fff8097c440 sp 0x7fff8097c430
READ of size 4 at 0x602000000010 thread T0
    #0 0x560f0c29f225 in main (/home/lab6/share/free_ASan+0x1225)
    #1 0x7f066f9990b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)
    #2 0x560f0c29f10d in _start (/home/lab6/share/free_ASan+0x110d)

0x602000000012 is located 0 bytes to the right of 2-byte region
[0x602000000010,0x602000000012)
freed by thread T0 here:
    #0 0x7f066fc7440f in __interceptor_free
../../../../src/libsanitizer/asan/asan_malloc_linux.cc:122
    #1 0x560f0c29f1ee in main (/home/lab6/share/free_ASan+0x11ee)
    #2 0x7f066f9990b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)

previously allocated by thread T0 here:
    #0 0x7f066fc74808 in __interceptor_malloc
../../../../src/libsanitizer/asan/asan_malloc_linux.cc:144
    #1 0x560f0c29f1de in main (/home/lab6/share/free_ASan+0x11de)
    #2 0x7f066f9990b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)

SUMMARY: AddressSanitizer: heap-use-after-free (/home/lab6/share/free_ASan+0x1225)
in main
Shadow bytes around the buggy address:
  0x0c047fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c047fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c047fff8000: fa fa[fd]fa fa fa fa fa fa fa fa fa fa fa fa fa
```

```
  0x0c047fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff8050: 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
==688==ABORTING
```

valgrind report

```
==690== Memcheck, a memory error detector
==690== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==690== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==690== Command: ./free
==690==
==690== Invalid read of size 4
==690==    at 0x109193: main (in /home/lab6/share/free)
==690==  Address 0x4a42040 is 0 bytes inside a block of size 2 free'd
==690==    at 0x483CA3F: free (in /usr/lib/x86_64-linux-
gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==690==    by 0x10918E: main (in /home/lab6/share/free)
==690==  Block was alloc'd at
==690==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-
gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==690==    by 0x10917E: main (in /home/lab6/share/free)
==690==
==690==
==690== HEAP SUMMARY:
==690==     in use at exit: 0 bytes in 0 blocks
==690==   total heap usage: 1 allocs, 1 frees, 2 bytes allocated
==690==
==690== All heap blocks were freed -- no leaks are possible
```

```
==690==
==690== For lists of detected and suppressed errors, rerun with: -s
==690== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

ASan 能, valgrind 能

Use-after-return

code

```
char* x;

void foo() {
    char stack_buffer[42];
    x = &stack_buffer[13];
}

int main() {
    foo();
    *x = 42; // Boom!
    return 0;
}

// gcc UseAfterReturn.c -o return
// gcc UseAfterReturn.c -fsanitize=address -o return_ASa

// ASAN_OPTIONS=detect_stack_use_after_return=1 ./return_ASan
// valgrind ./return
```

ASan report

```
=================================================================
==59==ERROR: AddressSanitizer: stack-use-after-return on address 0x7f82ee765025 at
pc 0x55e927c742f3 bp 0x7ffee31ae200 sp 0x7ffee31ae1f0
WRITE of size 1 at 0x7f82ee765025 thread T0
    #0 0x55e927c742f2 in main (/home/lab6/share/return_ASan+0x12f2)
    #1 0x7f82f1f9b0b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)
    #2 0x55e927c7410d in _start (/home/lab6/share/return_ASan+0x110d)

Address 0x7f82ee765025 is located in stack of thread T0 at offset 37 in frame
    #0 0x55e927c741d8 in foo (/home/lab6/share/return_ASan+0x11d8)

  This frame has 1 object(s):
    [32, 42) 'stack_buffer' (line 4) <== Memory access at offset 37 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/lab6/share/return_ASan+0x12f2) in main
Shadow bytes around the buggy address:
  0x0ff0ddce49b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce49c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce49d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce49e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce49f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0ff0ddce4a00: f5 f5 f5 f5[f5]f5 f5 f5 00 00 00 00 00 00 00 00
  0x0ff0ddce4a10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce4a20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce4a30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce4a40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ff0ddce4a50: 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
==59==ABORTING
```

valgrind report

```
==61== Memcheck, a memory error detector
==61== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==61== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==61== Command: ./return
==61==
==61==
==61== HEAP SUMMARY:
==61==     in use at exit: 0 bytes in 0 blocks
==61==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==61==
==61== All heap blocks were freed -- no leaks are possible
==61==
==61== For lists of detected and suppressed errors, rerun with: -s
==61== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

ASan 能, valgrind 不能

Stack buffer overflow 剛好越過 redzone

code

```
int main(){
    int a[8] = {0};
    int b[8] = {0};

    // fail to detect
    int access = a[16];
    return 0;

    // successfully detected
    // int access = a[15];
    // 1-7 (8-15) 16-23 (24-31)
}

// gcc -fsanitize=address redzone.c -o redzone_ASan
// ./redzone_ASan
```

ASan 不能偵測出跨過 redzone 的讀寫。

TAGS: `SOFTWARE-TESTING`