Wykrywanie incydentów - lab 7

Zadanie 1

Dla `Example 1` Valgrind nie wykrywa żadnych błędów z alokacją pamięci

```
==66008== Memcheck, a memory error detector
==66008== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==66008== Using Valgrind-3.18.0.GIT-lbmacos and LibVEX; rerun with -h for
copyright info
==66008== Command: ./a.out
==66008==
==66008== Warning: set address range perms: large range [0x7fff20014000,
0x80001fe14000) (defined)
==66008== Warning: set address range perms: large range [0x7fff20014000,
0x7fff7fcdc000) (defined)
==66008== Warning: set address range perms: large range [0x7fff8e268000,
0x7fffc0014000) (noaccess)
==66008== Warning: set address range perms: large range [0x7fffc0014000,
0x7fffe2e34000) (defined)
==66008== Warning: set address range perms: large range [0x7fffe2e34000,
0x7fffffe00000) (noaccess)
==66008==
==66008== HEAP SUMMARY:
==66008==     in use at exit: 0 bytes in 0 blocks
==66008==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==66008==
==66008== All heap blocks were freed -- no leaks are possible
==66008==
==66008== For lists of detected and suppressed errors, rerun with: -s
==66008== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Dla `Example 2` Valgrind pokazuje więcej błędów, ale są one związane tylko z przypisywaniem wartości do niezainicjalizowanych elementów tablicy.

```
==66304== Memcheck, a memory error detector
==66304== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==66304== Using Valgrind-3.18.0.GIT-lbmacos and LibVEX; rerun with -h for
```

```
copyright info
==66304== Command: ./a.out
==66304==
==66304== Warning: set address range perms: large range [0x7fff20014000,
0x80001fe14000) (defined)
==66304== Warning: set address range perms: large range [0x7fff20014000,
0x7fff7fcdc000) (defined)
==66304== Warning: set address range perms: large range [0x7fff8e268000,
0x7fffc0014000) (noaccess)
==66304== Warning: set address range perms: large range [0x7fffc0014000,
0x7fffe2e34000) (defined)
==66304== Warning: set address range perms: large range [0x7fffe2e34000,
0x7fffffe00000) (noaccess)
==66304== Conditional jump or move depends on uninitialised value(s)
==66304==    at 0x7FFF203658DF: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20233194: ??? (in /dev/ttys002)
==66304==    by 0x7FFF2023BE3B: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20260974: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20238FF5: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20237165: ??? (in /dev/ttys002)
==66304==    by 0x100003F34: main (ex2.c:11)
==66304==
==66304== Conditional jump or move depends on uninitialised value(s)
==66304==    at 0x7FFF2023ADF0: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20260974: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20238FF5: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20237165: ??? (in /dev/ttys002)
==66304==    by 0x100003F34: main (ex2.c:11)
==66304==
==66304== Conditional jump or move depends on uninitialised value(s)
==66304==    at 0x7FFF2023D5CD: ??? (in /dev/ttys002)
==66304==    by 0x7FFF2023AF56: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20260974: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20238FF5: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20237165: ??? (in /dev/ttys002)
==66304==    by 0x100003F34: main (ex2.c:11)
==66304==
==66304== Syscall param write(buf) points to uninitialised byte(s)
==66304==    at 0x7FFF202F02BE: ??? (in /dev/ttys002)
```

```
==66304==    by 0x7FFF202380B8: ??? (in /dev/ttys002)
==66304==    by 0x7FFF202308F7: ??? (in /dev/ttys002)
==66304==    by 0x7FFF2023324D: ??? (in /dev/ttys002)
==66304==    by 0x7FFF2023D0CA: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20260974: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20238FF5: ??? (in /dev/ttys002)
==66304==    by 0x7FFF20237165: ??? (in /dev/ttys002)
==66304==    by 0x100003F50: main (ex2.c:13)
==66304==  Address 0x101008212 is in a rw- anonymous segment
==66304==
0 1 2 3 4 5 6 7 8 1
==66304==
==66304== HEAP SUMMARY:
==66304==     in use at exit: 0 bytes in 0 blocks
==66304==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==66304==
==66304== All heap blocks were freed -- no leaks are possible
==66304==
==66304== Use --track-origins=yes to see where uninitialised values come from
==66304== For lists of detected and suppressed errors, rerun with: -s
==66304== ERROR SUMMARY: 13 errors from 4 contexts (suppressed: 0 from 0)
```

Dla `Example 3` Valgrind ponownie nie komunikuje żadnych błędów.

```

==67445== Memcheck, a memory error detector
==67445== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==67445== Using Valgrind-3.18.0.GIT-lbmacos and LibVEX; rerun with -h for
copyright info
==67445== Command: ./a.out
==67445==
==67445== Warning: set address range perms: large range [0x7fff20014000,
0x80001fe14000) (defined)
==67445== Warning: set address range perms: large range [0x7fff20014000,
0x7fff7fcdc000) (defined)
==67445== Warning: set address range perms: large range [0x7fff8e268000,
0x7fffc0014000) (noaccess)
==67445== Warning: set address range perms: large range [0x7fffc0014000,
0x7fffe2e34000) (defined)
```

```
==67445== Warning: set address range perms: large range [0x7fffe2e34000,
0x7fffffe00000) (noaccess)
==67445==
==67445== HEAP SUMMARY:
==67445==     in use at exit: 0 bytes in 0 blocks
==67445==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==67445==
==67445== All heap blocks were freed -- no leaks are possible
==67445==
==67445== For lists of detected and suppressed errors, rerun with: -s
==67445== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Zadanie 2

Kod pliku nagłówkowego `SafeArray.h`

```
#include <stdexcept>

class SafeArray
{
private:
    int* array;
    int size;
public:
    SafeArray(int size);
    ~SafeArray();
    void set(int index, int element);
    int get(int index);
};
```

Kod klasy `SafeArray.cpp`

```
#include „SafeArray.h”

SafeArray::SafeArray(int size)
{
    SafeArray::size = size;
    array = new int[size];
}

SafeArray::~SafeArray()
{
    delete[] array;
}

void SafeArray::set(int index, int element)
{
    if (index < 0 || index >= size)
    {
        throw std::invalid_argument(„Index out of range.”);
    }
    array[index] = element;
}

int SafeArray::get(int index)
{
    if (index < 0 || index >= size)
    {
        throw std::invalid_argument(„Index out of range.”);
    }
    return array[index];
}
```

Kod funkcji `main.cpp`

```
#include <iostream>
#include „SafeArray.h”

int main() {
    auto* array = new SafeArray(10);
    array->set(5, 5);
    std::cout << „array->get(5): „ << array->get(5) << std::endl;
    std::cout << „array->get(4): „ << array->get(4) << std::endl;
    std::cout << „array->get(10): „ << array->get(10) << std::endl;
    return 0;
}
```

Wynik wykonania programu:

```
array->get(5): 5
array->get(4): 0
array->get(10): libc++abi: terminating with uncaught exception of type
std::invalid_argument: Index out of range.

Process finished with exit code 134 (interrupted by signal 6: SIGABRT)
```

Zadanie 3

Flaga `-fomit-frame-pointer` zapewnia omijanie wskaźnika ramki w funkcjach, które go nie potrzebują. Dzięki temu instrukcje nie zapisują, nie ustawiają ani nie odzyskują wskaźnika ramki.

Flaga `-fipa-cp` zapewnie interproceduralną propagację stałych. Ta optymalizacja analizuje program w celu określenia, kiedy wartości przekazywane do funkcji są stałymi i wówczas optymalizuje ten proces. Flaga szczególnie przydatna w przypadku gdy program posiada wiele funkcji, do których przekazywane są stałe.

Flaga `-fcf-protection=[full|branch|return|none]` uruchamia instrukcję do zwiększenia bezpieczeństwa programu przez sprawdzanie czy docelowe adresy instrukcji kontroli przepływu (takich jak niebezpośrednie wywołanie funkcji, wynik funkcji, niebezpośredni skok) są poprawne.