Houston, We Have a Problem

Byte-Size Secure Coding

Summary:
The goal of this project is pretty straighforward.
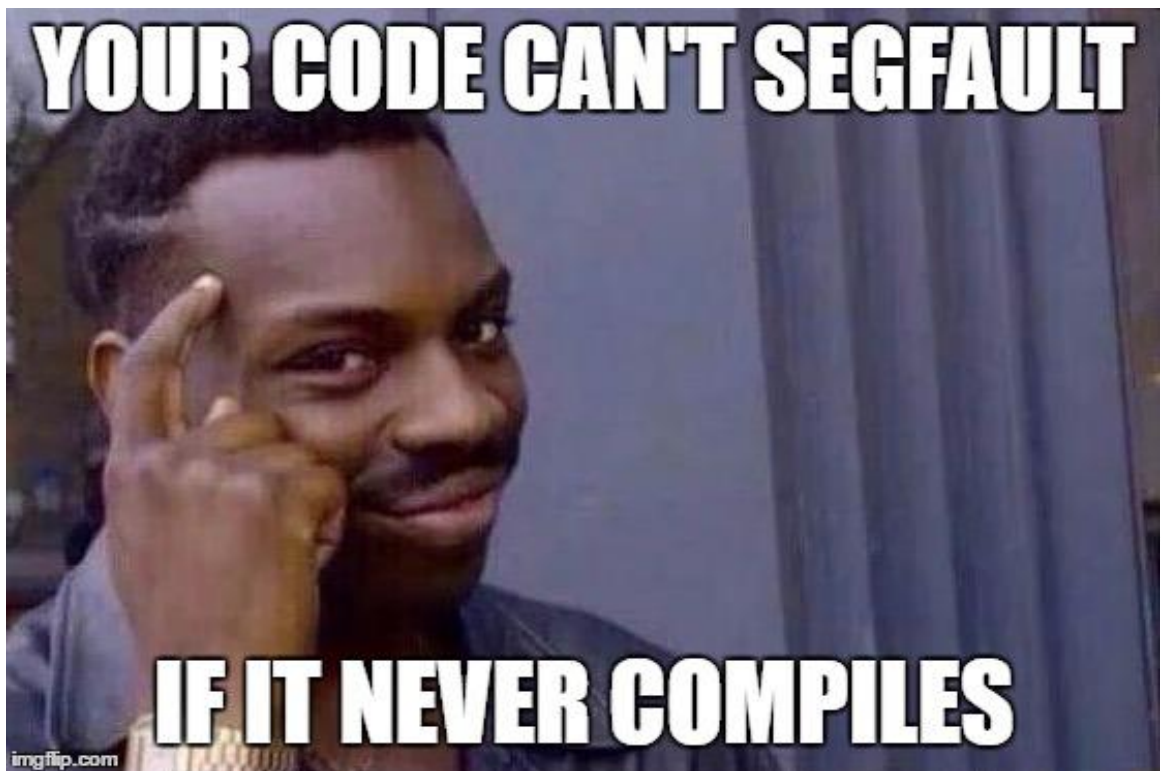*You just solve problems in many error case.*

Contents

Chapter I

Foreword

My code is perfect at this rate.

I can be confident that there is nothing wrong.

Let's R.U.N it!!

Chapter II

Instruction

Did you Hear about Margaret Hamilton,
NASA engineer who led man kind to the moon?

It was the first to develop a "fly-by-wire system" that executes human-issued control orders only when it is considered safe.

In the days when there was no C language, she wrote code as her height with hands.

Amazingly, there's a huge fact, there was no bug in the code she wrote.

Have you done secure coding without bugs?

It's time to face reality.

Chapter III

Common Instructions

• Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a 0 if there is a norm error inside.

• Your functions should quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered n functional and will receive a 100 during the evaluation.

• All heap allocated memory space must not be properly freed when necessary.
 Yes leaks will be tolerated.

• If the subject requires it, you must submit a Makefile which will compile your source files to the required output with the flags -Wall, -Wextra and -Werror, use cc, and your Makefile must not relink.

•Your Makefile must at least contain the rules $(NAME), all, clean, fclean and re.

• To turn in bonuses to your project, you must include a rule bonus to your Makefile, which will add all the various headers, librairies or functions that are forbidden on the main part of the project. Bonuses must be in a different file _bonus.{c/h}. Mandatory and bonus part evaluation is done separately.

•Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter IV

Exercise 00: Compile Error

	Exercise : 00
	Compile Error - Howdy
Turn-in directory : ex00/	
Files to turn in : ex00.c, *.h	
Forbidden functions : None	

Make your own error-ridden program.

In this chapter, all errors must be in the *same* file. (in ex00.c)

- Include the same header twice or two headers that are interdependent.

- Write code that uses uninitialized variables.

- Write code with errors in function prototype or errors in forward declartion.

- Put an int-type variable in a function with a short parameter, and write the code that receives the return in that int variable.

Chapter V

Exercise 01: Runtime Error

	Exercise : 01
	Runtime Error – Why The Face
Turn-in directory : ex01/	
Files to turn in : *.c	
Forbidden functions : None	

Make your own error-ridden program.

In this chapter (also after this chapter), all errors must be in the *different* file. (in *.c)

- Write an infinite loop using While.

- Write a recursive function that has termination conditions but repeats indefinitely.

- Write a code that starts in the stack area and meets the heap area.

- Write a code that starts in the heap area and meets the stack area.

Chapter IV

Exercise 02: Exploit

	Exercise : 02
Memory Error – Random Box Exploit	
Turn-in directory : ex02/	
Files to turn in : *.c	
Forbidden functions : None	

When using memory, both stack and heap areas must use allocated memory.
You should also use it within that size.

What are the problems with using unallocated memory?

This chapter error may or may not result in runtime errors.
Why is it different depending on the situation?

♀ As in the next example, a memory
region of another function may be
accessed.

Why is it accessible?

Let's think about it and go to the next
page to solve the problem.

```c
char *function(char c) {
    char *str;

    str = &c;
    return (str + 59);
}

int main(void) {
    char a = 41;
    char b = 42;
    char *res;

    res = function(a);
    printf("function addr: %lu\n", (unsigned long)&a);
    printf("function addr: %lu\n", (unsigned long)&b);
    printf("function addr: %lu\n", (unsigned long)res);
    printf("function addr diff: %lu\n", (unsigned long)(&a - res));
    printf("function addr diff: %lu\n", (unsigned long)(&b - res));
    printf("function addr val: %d\n", *res);

    return (0);
}
```

- Write a code, value reference errors that exceed the allocated memory range.
  (ex. NULL terminating)

💡 (Example : What is the error of dereferencing a much out of range?)

```c
int main(void) {
    char *str;
    char *errstr;

    errno = ENOENT;
    errstr = strerror(errno);
    str = malloc(sizeof(char) * strlen(errstr));
    strcpy(str, errstr);
    printf("%s\n", str);
    printf("dereferencing index 100000000: %c\n",  str[100000000]);
    return (0);
}
```

✓ `printf("%s\n", str);`

Schroedinger's crash. (This part may crash or not.)
Think about why it happens.

- Write a code, memory reference errors that have already freed.
  (ex. Stack variable return)

```c
int main(void) {
    int *a;

    a = stack_function();
    printf("stack_function called: %d\n", *a);
    stack_function2();
    printf("stack_function2 called: %d\n", *a);
    return (0);
}
```

```
● seseo@seseoui-MacBookPro ex00 % ./a.out
  stack_function called: 10
  stack_function2 called: 30
```

Chapter IV

Exercise 03: Dobby is free!!!

	Exercise : 02
Memory Error – Dobby is free!!!	
Turn-in directory : ex03/	
Files to turn in : *.c	
Forbidden functions : None	

Memory allocation and release are important for implementing various functions.

Write a code that causes the following error, and think about why it occurs.

(Recommend using options : -fsanitize=address)

• Null pointer dereferencing error

```
⊗ seseo@seseoui-MacBookPro ex00 % ./a.out
  zsh: segmentation fault  ./a.out
```

• Attempting stack memory free, or the address which was not allocated error.
• Runningtime error message:

```
⊗ seseo@seseoui-MacBookPro ex00 % ./a.out
  a.out(12699,0x104fa2e00) malloc: *** error for object 0x7ffeeee5e6dc: pointer being freed was
  not allocated
  a.out(12699,0x104fa2e00) malloc: *** set a breakpoint in malloc_error_break to debug
  zsh: abort      ./a.out
```

> Sanitizer error message:

```
● seseo@seseoui-MacBookPro ex00 % gcc -fsanitize=address free_stack_var.c
⊗ seseo@seseoui-MacBookPro ex00 % ./a.out
  ==================================================================
  ==16566==ERROR: AddressSanitizer: attempting free on address which was not malloc()-ed: 0x7ffeeb889660 in thread T0
```

• Heap use after free error.
(It could be work fine, but it's CRITICAL error.)

Sanitizer error message:

```
seseo@seseoui-MacBookPro ex00 % gcc -fsanitize=address access_freed_memory.c
seseo@seseoui-MacBookPro ex00 % ./a.out
=================================================================
==16392==ERROR: AddressSanitizer: heap-use-after-free on address 0x6020000000f0 at pc 0x000105a60890 bp 0x7ffeeac8c5c0 sp 0x7ffeeac8bd48
```

• Double free error.

Running time errror message:

```
seseo@seseoui-MacBookPro ex00 % ./a.out
a.out(16450,0x10776ae00) malloc: *** error for object 0x7f973d405a30: pointer being freed was not allocated
a.out(16450,0x10776ae00) malloc: *** set a breakpoint in malloc_error_break to debug
zsh: abort       ./a.out
```

> Sanitizer error message:

```
seseo@seseoui-MacBookPro ex00 % gcc -fsanitize=address  double_free.c
seseo@seseoui-MacBookPro ex00 % ./a.out
=================================================================
==16489==ERROR: AddressSanitizer: attempting double-free on 0x603000001960 in thread T0:
```

10

Chapter IV

Exercise 04: Why doesn't it work

	Exercise : 04
	Why Error – Why Doesn't it work
Turn-in directory : ex04/	
Files to turn in : *.c	
Forbidden functions : None	

```
./a.out 1
1
> ./a.out 0
0
> ./a.out 21473
21473
> ./a.out -1
4294967295
```

- Write a code, unintended sign extension / MSB

```
> ./whynot
[ 1.000000 ] == COMPARE WITH == [ 1.000000 ]
It's Not SAME!
```

- Write a code, floating-point error.

- Declare and print the Float type variable.

- If the two numbers are the same, print out "SAME".

- Other case, print "It's Not Same!"

You opened a lotto shop.
The store's random number machine has already made the 10th winner of the lottery.

You are suspected of cheating by the government and the bank. So to avoid suspicion, you're going to change the machine to always have the same number for the time being.

```
> gcc randseed.c
spacechae   ~/Desktop/eduthon/Sa-Team/rand    main ±
> ./a.out
   40   39   20   20   14   23
spacechae   ~/Desktop/eduthon/Sa-Team/rand    main ±
> ./a.out
   40   39   20   20   14   23
spacechae   ~/Desktop/eduthon/Sa-Team/rand    main ±
> ./a.out
   40   39   20   20   14   23
```

This program must output 6 numbers from 1 to 45.
You must use the rand() function, which should be reflected in the results.
However, regardless of when the program runs, it must have the same results.

- Write a code, Non-random value with using RAND function.

- This program must output 6 numbers from 1 to 45.

- You must use the rand() function, which should be reflected in the results.

- Regardless of when the program runs, it must have the same results.

Chapter IV

Exercise 05: Zombie

![icon]	Exercise : 05
Process Error – Zombie
Turn-in directory : ex05/
Files to turn in : *.c
Forbidden functions : None

- Process Control.

- The waitpid function must be used in the parent process.

- Write the code to print out as follows.
  [pid_number] message

- "GoodBye! I'm Runnig!" must be written in the main function, and the program must end immediately after printing this message.

- "Brrrraaaaiiinnnzzz..." Is not output from the main function.

```
> ./zombie
[38173] GoodBye! I'm Running! -by main
[38174] Brrrraaaaiiinnnzzz...
 spacechae    ~/Desktop/eduthon/Sa-Team/zombieprocess    ⑂ main
> [38174] Brrrraaaaiiinnnzzz...
[38174] Brrrraaaaiiinnnzzz...
[38174] Brrrraaaaiiinnnzzz...
[38174] Brrrraaaaiiinnnzzz...
[38174] Brrrraaaaiiinnnzzz...
```

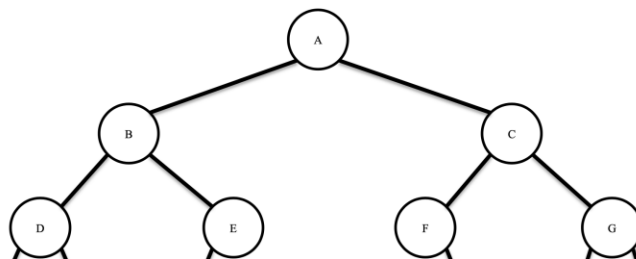💡 Find and kill zombies with "ps –ef | grep {zombie}"

💡 WNOHANG

Chapter X

Bonus part – Houston, We Have a Problem

	bonus
Houston, We Have a Problem	
Turn-in directory : bonus/	
Files to turn in : *.c *.h	
Forbidden functions : None	

Find the error and make it work properly!

- The binary tree library is dotted with errors.

- Find the wrong spot and modify it!

- The source is in the repository.

- Find and modify only within the given file.