INF226 Assignment 1 - Buffer Overflow

Willem Schooltink - Håkon Gylterud

August 2024

This is the first of the three mandatory assignments of INF226. In this exercise you will be exploring buffer overflows and program memory. Your goal for each exercise is to read the contents of `flag.txt`. This project is due **Friday, September 13th**, 23:59, and will count 10% of your final grade.

You can collaborate on these exercises in groups of up to three students. Find an available group on MittUiB and remember that every member must sign up to the group.

Deliverables

For this exercise you deliver one report in PDF format. The report should contain the following, for each exercise:

1. a brief description of the vulnerability,
2. a description of how you exploit the vulnerability,
3. the code you used to exploit the vulnerability,
4. the string found in `flag.txt`.

Practicals

Each exercise is hosted online at `oblig1.bufferoverflow.no` at the port specified in the exercise description. For each exercise you will be given the source code, and the compiled ELF 64-bit binary file that is running on the server. You can try to solve the exercises locally first, but the final solution should work on the remote server.

If you get stuck and don't manage to complete an exercise, be sure to still write down what you have found. You may still get points for partial solutions!

All binaries have been compiled using the following command:

```
gcc #.c -o # -w -fstack-protector -no-pie,
```

with `#` being the number of the exercise.

Exercise 1. (3pts) - Getting started

Exercise 1 is available on the remote server on port `7001`. The files for this exercise are found in folder '1', with relevant files '1.c' and '1'.

Source Code - 1.c

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

int main(int argc, char *argv[]) {
    struct locals {
        char buffer[16];
        int32_t secret;
    } locals;
    locals.secret = 0x4841434b;

    printf("A new challenger approaches!\n");
    fflush(stdout);
    assert(fgets(locals.buffer, 1024, stdin) != NULL);

    if (locals.secret == 0xc0ffee) {
        printf("You found the flag!\n");
        fflush(stdout);
        system("cat flag.txt");
    } else {
        printf("No flag here\n");
        fflush(stdout);
    }

    return 0;
}
```

Exercise 2. (3pts) - Ramping Up

Exercise 2 is available on the remote server on port `7002`. The files for this exercise are found in folder '2', with relevant files '2.c' and '2'.

Source Code - 2.c

```c
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>

void expose_flag() {
    system("cat flag.txt");
}

char* pick_animal() {
    // Array of predetermined strings
    const char *options[] = {
        "Dog",
        "Cat",
        "Giraffe",
        "Sea Horse",
        "Gremlin",
        "Snake",
        "Penguin"
    };

    int num_options = sizeof(options) / sizeof(options[0]);
    srand(time(NULL));
    int random_index = rand() % num_options;

    return options[random_index];
}

int main(int argc, char *argv[]) {
    struct locals {
        char buffer[32];
        char* (*func_pt)();
    } locals;
    locals.func_pt = pick_animal;

    printf("You know what, I'm curious: what is you favourite
    animal?\n");
    fflush(stdout);

    assert(fgets(locals.buffer, 1024, stdin) != NULL);

    char* animal = locals.func_pt();
    printf("Really? My favourite animal would be a %s\n", animal);

    return 0;
}
```

Exercise 3. (2.5pts) - Pretty Tricky

Exercise 3 is available on the remote server on port `7003`. The files for this exercise are found in folder '3', with relevant files '3.c' and '3'.

Source Code - 3.c

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

void expose_flag() {
    system("cat flag.txt");
}

int main(int argc, char *argv[]) {
    char buffer[16];

    printf("Welcome explorer, to this wonderous world.\n");
    fflush(stdout);

    int exploration_offset;

    scanf("%d", &exploration_offset);
    getchar();

    printf("You expolore and find %lx\n",
            *(unsigned long*)(buffer + exploration_offset));

    printf("Now, try to find the flag!\n");
    fflush(stdout);

    assert(fgets(buffer, 1024, stdin) != NULL);

    return 0;
}
```

Exercise 4. (1.5pts) - Also Quite Tricky

Exercise 4 is available on the remote server on port `7004`. The files for this exercise are found in folder '4', with relevant files '4.c' and '4'.
Tip: by default the first few arguments of a function are passed by register, meaning that these arguments will not show up in the stack. In the case of our server the first 6 arguments are passed by register.

Source Code - 4.c

```c
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

struct wrapper {
    char buffer[16];
    char* identifier;
};

void program(int arg1, int arg2, int arg3, int arg4, int arg5, int arg6, char* secret) {
    struct wrapper input_wrapper;
    input_wrapper.identifier = "Guest";

    printf("Weclome, you have access to this program as: %s\n", input_wrapper.identifier);
    printf("What is your favourite number?\n");
    fflush(stdout);

    int offset = 0;
    scanf("%d", &offset);
    getchar();

    printf("My secret message for your number is:\n");
    printf("%lx\n", *(unsigned long*)(input_wrapper.buffer + offset));

    printf("Do you want to know my secret?\n");
    fflush(stdout);

    assert(fgets(input_wrapper.buffer, 1024, stdin) != NULL);

    if (strcmp(input_wrapper.identifier, secret) == 0) {
        printf("My secret is:\n");
        fflush(stdout);
        system("cat flag.txt");
    } else {
        printf("Well, I wont tell my secret to just any guest.\n");
        fflush(stdout);
    }
}

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

```
43      if (argc < 2) {
44          fprintf(stderr, "Usage: %s <secret>\n", argv[0]);
45          return EXIT_FAILURE;
46      }
47
48      program(1,2,3,4,5,6,argv[1]);
49      return 0;
50  }
```