# Computer Security (COM-301)

## Monday live exercises

## Execution Attacks and Security testing

**Carmela Troncoso**

SPRING Lab

carmela.troncoso@epfl.ch

# DEP protection

The startup `NewHaven` hires you to help them develop secure software.

The `NewHaven` developers tell you that in their server, they sometimes need to execute code that is provided as a parameter in a function. But they tell you that this is safe because the server implements Data Execution Prevention. Is this correct? (Justify your answer).

# The (true or not) power of the canary

(a) Stack canaries, which are a defense against buffer overflow attacks, require operating system support

(a) Stack canaries will prevent any attack that overflows local variables from executing injected code

(a) Stack canaries protect against all printf format string vulnerabilities

# Bypassing the password

Consider the following C function for getting a password from the standard input and checking it (strcmp compares two strings, and returns 0 if strings are equal, or a number that is not equal to zero if they are not):

```
getPassword() {
    int isCorrect = 0;
    char password[12];

    printf("Enter your password > ");
    gets(password);

    if(strcmp(password, "C0m301-Adm1N") == 0) isCorrect = 1;

    return isCorrect;
}
```

Consider the following fuzzing strategy: a fuzzer generates and feeds strings composed of multiple '\0' characters (null string terminators), of length up to 15.
Is this strategy able to uncover a vulnerability? If yes, identify which vulnerability and explain why the fuzzer finds it. If not, propose an alternative approach to test (may or may not be fuzzing) that uncovers a vulnerability. Explain why your approach would find that vulnerability.

# Bonus for everyone

Gru has written the code in the next slide to manage the bonuses of minions that participate in evil missions. This function receives an identifier for a minion and then prompts the minion to provide the name of the mission they participated in. Gru asks for your help debugging the function.

Identify two lines that contain unsafe code that may lead to a memory safety error. For those two lines i) explain the vulnerability, and ii) explain whether a Minion can exploit this vulnerability to increase their bonus even when their mission did not succeed.

*Assume that Minions are good teammates and will never steal a bonus from another Minion by providing others' successful mission names*

*Assume that local variables are pushed to the pile as in the order they are created*

# Bonus for everyone

Identify two lines that contain unsafe code that may lead to a memory safety error. For those two lines i) explain the vulnerability, and ii) explain whether a Minion can exploit this vulnerability to increase their bonus even when their mission did not succeed.

```
int bonus[100] =  { 0 };                    /* array of 100 integers initialized to 0 */

char successDB(char* mission) {

    /* function that returns 0 for failed missions */
    /* and 1 for successful missions */
    /* if the mission does not exist, it crashes */

}

1: int AddMission(int minionID) {
2:      char success;                       /* one byte: 1 if mission succeeds, 0 otherwise */
3:      char mission[30];                   /* mission name */
4:
5:      gets(mission);                      /* reads mission name from keyboard */
6:      success = successDB(mission);       /* checks in the DB the success of the mission */
7:
8:      if (success == 1) { bonus[minionID] += 1 }; /* if mission succeeded, increase bonus */
9:
10:     return 0;
11: }
```

# The more fuzz, the fuzzier

When you fuzz a program… True or false (as always, justify!)

[a] Using one fuzzer is enough

[b] You should use as many fuzzers as possible for as long as possible

[c] Running three complementary fuzzers for 1000 iterations each is enough

[d] It does not matter which fuzzer you choose. All types are good.