Foundations of Cybersecurity C/C++ Secure Coding Basics Laboratory Exercises

Michele La Manna
Dept. of Information Engineering
University of Pisa
michele.lamanna@phd.unipi.it
Version: 2021-03-18

The fool, lazy colleague

In your company there is this dude, super funny but lazy and a fool.

He knows how to put together some lines of code, however, he completely neglects the danger of cyber attacks, malicious users, or monkey users.

{Edit. *monkey users:* the kind of person who puts a metal spoon in a microwave because he/she saw a lifehack on twitter}

Your colleague wrote some terrible functions, and you need to understand:

1. What he wanted to do (total absence of comments);
2. Analyze the code and detect vulnerabilities;
3. Patch the vulnerability

For some reasons, you can't change some lines, and you also MUST NOT change the main()!

Requests, tips&tricks

UNIVERSITÀ DI PISA

The moment after I will give you an exercise, I will write in chat something like "Ready" "I got it" "Done" etc…
Please react to that message **only when you solved the exercise.**
In this way I can learn how much time most of the class needs on those kind of assignments.

In these first laboratories I suggest you use https://www.onlinegdb.com/online_c++_compiler to test the "lazy colleague's code" and patch it.

Sometimes I will ask questions to some of you, please don't panic!
It is only a way to create engagement in the subject, not to humiliate you. If you prefer not to speak out loud, you will not be forced to.

TUTORIAL

DoStuff

UNIVERSITÀ DI PISA

```
35  int main() {
36      string str;
37      size_t offset;
38
39      cout << "Welcome to do_stuff() invoker." << endl;
40      cout << endl;
41
42      for(;;){
43          cout << "str = ";
44          cin >> str;
45          if(!cin) return 0;
46          cout << "offset = ";
47          cin >> offset;
48          if(!cin) return 0;
49          cin.ignore(numeric_limits<streamsize>::max(), '\n'); // clear cin buffer
50          cout << "Invoking do_stuff(" << str << "," << offset << ")..." << endl;
51          do_stuff(str.c_str(), offset);
52          cout << endl;
53      }
54      return 0;
55  }
56
```

DoStuff Part#1

UNIVERSITÀ DI PISA

In this function, the parameter «str» must be considered as tainted.
For simplicity, we say that the tainted string is also well-formed (it has a terminator, it does not contain problematic characters etc…).

```
8   void do_stuff(const char* str, size_t offset){
9    // NOTE: For esoteric reasons, C++ strings cannot be
10       // PART #1
11       char result1[100]; // UNCHANGEABLE!
12       strcpy(result1, str);
```

Your dear colleague wanted to make a copy of the string str.
However, this operation can lead to a buffer overflow!
Indeed, if the size of the parameter str is greater than 100, the strcopy goes beyond the buffer until it reaches the terminator ('\0').

DoStuff Solution#1

UNIVERSITÀ DI PISA

We cannot change the buffer dimension, therefore we have to use the function «str**n**copy». In this way, you can force a limit on the number of character to copy inside the buffer.

```
8   void do_stuff(const char* str, size_t offset){
9   // NOTE: For esoteric reasons, C++ strings cannot
10      // PART #1
11      char result1[100]; // UNCHANGEABLE!
12      strncpy(result1, str, 100);
13      result1[99] = '\0';
```

CAREFUL! The function «strncopy» DOES NOT put a terminator at the last position.
You must remember to manually insert it at the last possible position.
https://www.cplusplus.com/reference/cstring/strncpy/?kw=strncpy

DO STUFF#2

DoStuff Part#2

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
14    // PART #2
15    char result2[50]; // UNCHANGEABLE!
16    memset(result2, 'a', offset);
17    printf("[do_stuff()] Please type the second part of result2: ");
18    if (gets(result2 + offset) == NULL) return;
```

10 minutes!

DoStuff Part#2

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
14    // PART #2
15    char result2[50]; // UNCHANGEABLE!
16    memset(result2, 'a', offset);
17    printf("[do_stuff()] Please type the second part of result2: ");
18    if (gets(result2 + offset) == NULL) return;
```

The lazy programmer wants to write the character "a" on the first "offset" bytes of result2.
Then, the user fills the string with some more data from the keyboard.

You have 15 minutes to find and patch the vulnerabilities.

DoStuff Solution#2

UNIVERSITÀ DI PISA

First of all, what if the tainted parameter "offset" is greater than 50?
We can either make the function return an error code or we can sanitize the offset. https://www.cplusplus.com/reference/cstdio/fgets/

```
15    // PART #2
16    char result2[50]; // UNCHANGEABLE!
17    if (offset > 49) offset = 49;
18    memset(result2, 'a', offset);
19    printf("[do_stuff()] Please type the second part of result2: ");
20    if(fgets(result2 + offset, 50 - offset, stdin) == NULL) return;
21    char* p = strchr(result2 + offset, '\n');
22    if (p) { *p = '\0'; }
```

Moreover, remember that the function «fgets» stops reading at either «\n» or «\0».
The fgets, however, automatically appends the "\0" terminator at the end of the string. Lines 21-22 are **recommended** if you are working with strings acquired from different sources and you want to uniform them (which means all strings must be terminated with only '\0' and have no special characters in them).

DO STUFF#3

DoStuff Part#3

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
20    // PART #3
21    char result3[60]; // UNCHANGEABLE!
22    strcpy(result3, "abcdefghij");
23    printf("[do_stuff()] Please type the second part of result3: ");
24    scanf("%s", result3 + 10); // for esoteric reasons, nothing except scanf() can be used here
```

10 minutes!

DoStuff Part#3

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
20    // PART #3
21    char result3[60]; // UNCHANGEABLE!
22    strcpy(result3, "abcdefghij");
23    printf("[do_stuff()] Please type the second part of result3: ");
24    scanf("%s", result3 + 10); // for esoteric reasons, nothing except scanf() can be used here
```

The lazy programmer wants to prepend the string “abcdefghij” to an input string given by the user.

You have 5 minutes to find and patch the vulnerabilities.

DoStuff Solution#3

This time, the first part of the string is hard-coded and therefore of fixed length.
However, there still is some risk of buffer overflow.
For example, if at line 30 the user would have inserted a 50+ character long string, the scanf could have written out of the "result3" bounds.

```
26    // PART #3
27    char result3[60]; // UNCHANGEABLE!
28    strcpy(result3, "abcdefghij");
29    printf("[do_stuff()] Please type the second part of result3: ");
30    scanf("%49s", result3 + 10); // for esoteric reasons, nothing except scanf() can be used here
```

To mitigate this problem, we fix the maximum number of characters that the "scanf" can read, through the use of the **sub-specifier "width".**

https://www.cplusplus.com/reference/cstdio/scanf/?kw=scanf

DO STUFF#4

DoStuff Part#4

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
26    // PART #4
27    char result4[40]; // UNCHANGEABLE!
28    strcpy(result4, "ABCDEFGHIJ");
29    sprintf(result4 + 10, "--%s--", str); // for esoteric reasons, nothing except sprintf() can be used here
30
```

10 minutes!

DoStuff Part#4

UNIVERSITÀ DI PISA

What is the objective of the following code?
Figure it out!

```
26    // PART #4
27    char result4[40]; // UNCHANGEABLE!
28    strcpy(result4, "ABCDEFGHIJ");
29    sprintf(result4 + 10, "--%s--", str); // for esoteric reasons, nothing except sprintf() can be used here
30
```

The lazy programmer wants to prepend the string “ABCDEFGHIJ” to the input string given by the user at the beginning of the main() function.

You have 10 minutes to find and patch the vulnerabilities.

DoStuff Part#4

UNIVERSITÀ DI PISA

How many of you applied this solution?

```
32      // PART #4
33      char result4[40]; // UNCHANGEABLE!
34      strcpy(result4, "ABCDEFGHIJ");
35      sprintf(result4 + 10, "--%25s--", str);
```

DoStuff Solution#4

UNIVERSITÀ DI PISA

The solution seems identical to the DoStuff#3 problem, right?
WRONG!
Although the problem is technically the same, the used functions are different!

```
32      // PART #4
33      char result4[40]; // UNCHANGEABLE!
34      strcpy(result4, "ABCDEFGHIJ");
35      sprintf(result4 + 10, "--%.25s--", str);
```

The "sprintf" function follows the format string rules of the "printf" family.
The **width sub-specifier**, to the "sprintf" indicates the **minimum number of characters to be printed.**
To fix the **maximum number of string's character** to be printed, you have to use the **precision sub-specifier.**

https://www.cplusplus.com/reference/cstdio/sprintf/

DoStuff Part#4

UNIVERSITÀ DI PISA

width	description
(number)	Minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces. The value is not truncated even if the result is larger.
*	The *width* is not specified in the *format* string, but as an additional integer value argument preceding the argument that has to be formatted.

.precision	description
.number	For integer specifiers (`d`, `i`, `o`, `u`, `x`, `X`): *precision* specifies the minimum number of digits to be written. If the value to be written is shorter than this number, the result is padded with leading zeros. The value is not truncated even if the result is longer. A *precision* of `0` means that no character is written for the value `0`. For `a`, `A`, `e`, `E`, `f` and `F` specifiers: this is the number of digits to be printed **after** the decimal point (by default, this is 6). For `g` and `G` specifiers: This is the maximum number of significant digits to be printed. For `s`: this is the maximum number of characters to be printed. By default all characters are printed until the ending null character is encountered. If the period is specified without an explicit value for *precision*, `0` is assumed.
.*	The *precision* is not specified in the *format* string, but as an additional integer value argument preceding the argument that has to be formatted.

From: https://www.cplusplus.com/reference/cstdio/printf/

Laboratory Material

UNIVERSITÀ DI PISA

Shortly after each laboratory, the slides and the solutions will be uploaded to the "Files" Tab of our MS Teams "General" channel.

See you next week!