Secure Coding – Laboratory Exercises

Pericle Perazzo, PhD

Version 2018-03-13

Laboratory Exercises

- Download assignments and code at: http://www.iet.unipi.it/p.perazzo/teaching/index.html

Exercise #1 – Arrays

```
int insert_table(size_t pos, int value)
```

- Inserts the integer value <value> inside a global table at position <pos> (zero-based indexed)
- Increases the table capacity if needed
- Returns 0 on success, -1 on failure
- Initially, the global table has zero capacity

- The main program repeatedly:
  - asks user for argument values
  - and calls insert_table() with such values

Exercise #1 – Arrays

- insert_in_table() contains vulnerabilities
- Mount the following attacks:

1. Cause an out-of-bound write (segmentation fault) with a single call
   - Correct the flaw
2. Cause an out-of-bound write (segmentation fault) with two calls
   - Correct the flaw
3. Cause an out-of-bound write (segmentation fault) with two calls with another tactic
   - Correct the flaw

Exercise #2 – Arrays

```
long* create_long_array(int a, int b, int c)
```

- Allocates an array of <a> long integers, all of which are 0 except those from <b> to <c> indexes (zero-based) included which are -1

1. Implement the above function in secure coding
   - Use malloc() to allocate
   - Use memset() to write 0's and -1's

```
memset(void* ptr, 0x00, size_t num)
```

   - Writes <num> bytes to 0x00 beginning from <ptr>
   - A long integer = 0 is represented by sizeof(long) bytes = 0x00

```
memset(void* ptr, 0xFF, size_t num)
```

   - Writes <num> bytes to 0xFF beginning from <ptr>
   - A long integer = -1 is represented by sizeof(long) bytes = 0xFF

Exercise #3 – Strings

```
int create_file_list(const char* name)
```

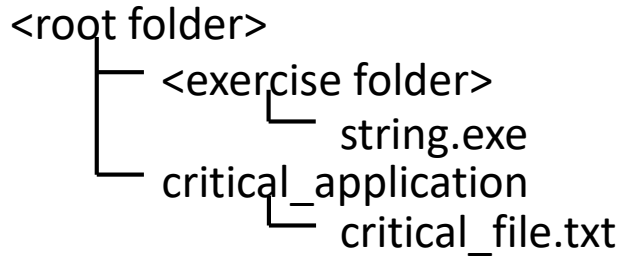
- Creates a text file named «<name>.txt» containing the result of the command «dir» (note: «dir» command is present in both Windows and Unix platforms)
- Returns 0 on success, -1 on failure

```
Il volume nell'unit… C non ha etichetta.
Numero di serie del volume: BA88-A5EA

Directory di C:\Users\Pericle\Documents\MEGAsync\Teaching\Seminario IDS\code\string

11/01/2018  13:45    <DIR>          .
11/01/2018  13:45    <DIR>          ..
11/01/2018  13:42                94 creditcards
11/01/2018  08:31                22 creditcards.bak
11/01/2018  13:45                 0 list.txt
11/01/2018  13:38             1.560 string.cpp
11/01/2018  13:38         1.566.073 string.exe
11/01/2018  13:38             4.250 string.o
11/01/2018  13:44             1.509 string_attacks.txt
10/01/2018  23:22             1.016 string_secure.cpp
               8 File      1.574.524 byte
               2 Directory  944.408.940.544 byte disponibili
```

Exercise #3 – Strings

- create_file_list() contains vulnerabilities
- Mount the following attacks:

1. Cause a buffer overflow
   - Correct the flaw
2. Cause the overwrite of the file named «critical_file.txt» in the folder named «critical_application», which is parallel to the current folder



3. Steal the content of the file named «creditcards»
   - Correct flaws 2 and 3

Exercise #4 – C++ Strings

```
int set_TMP_envvar(const std::string& name, const std::string& value)
```

- Assigns to an environment variable named «TMP_<name>» the value <value>
- Returns 0 on success, -1 on failure

```
std::string export_TMP_envvar(const std::string& name)
```

- Returns a string containing the name and the value of the environment variable named «TMP_<name>», with the following format:

  «Name:TMP_foo;Value:bar»

Exercise #4 – C++ Strings

- export_TMP_envvar() contains vulnerabilities
- Mount the following attack:

1. Cause an abnormal program termination (segmentation fault) with a single call of set_TMP_envvar() and export_TMP_envvar()
   - Correct the flaw

Exercise #5 – Unsigned Integers

```
char* create_string(size_t num1, size_t num2, char fillchar1, char fillchar2)
```

- Allocates a C string of length <num1>+<num2>, in which the first <num1> chars are <fillchar1>, and the following <num2> chars are <fillchar2>
- Example:

  create_string(3, 4, 'a', 'b') -> "aaabbbb"
- Returns a pointer to the string on success, NULL on failure

Exercise #5 – Unsigned Integers

- create_string() contains vulnerabilities
- Mount the following attack:

1. Cause an out-of-bound write (segmentation fault) with a single call
   - Correct the flaw

Exercise #6 – Unsigned Integers

```
int* create_int_array(size_t num, int fillint)
```

- Allocates an array of <num> integers, all with value <fillint>
- Returns a pointer to the array on success, NULL on failure

Exercise #6 – Unsigned Integers

- create_int_array() contains vulnerabilities
- Mount the following attack:

1. Cause an out-of-bound write (segmentation fault) with a single call
   - Correct the flaw

Exercise #7 – Signed Integers

```
void show_secret_information(int privilege)
```

- Prints on standard output the content of the file «secret_information.txt», only if <privilege> >= 100
- The <privilege> argument is a signed integer representing the user's privilege
- Negative privileges are meaningful

- The main program repeatedly:
  - asks user for his/her privilege,
  - replaces it with 99 if the user inserted >= 100,
  - and then calls show_secret_information()

Exercise #7 – Signed Integers

- show_secret_information() contains vulnerabilities
- Mount the following attack:

1. Gain unauthorized access to the secret information
   - Correct the flaw

Exercise #8 – Signed Integers

```
char* create_string2(int num1, int num2, char fillchar1, char fillchar2)
```

- Allocates a C string of length <num1>*<num2>, in which all the characters are <fillchar1>, except the 1st, the <num2>-th, the 2<num2>-th, and so on, which are <fillchar2>
- Example:

  create_string2(3, 4, 'a', 'b') -> "baaabaaabaaa"
- Returns a pointer to the string on success, NULL on failure

Exercise #8 – Signed Integers

- create_string2() contains vulnerabilities
- Mount the following attack:

1. Cause an out-of-bound write (segmentation fault) with a single call in several ways
   - Correct the flaw