C Programming Warm-up

Program One
CS 4471 Fall 2014

Due Dates: Thursday, Sep. 18, 11:59pm

1 Motivation

Many attacks on computer systems evolve from an attacker studying application source code to find a way to execute the code in a manner that the coder did not anticipate. Buffer overflows are a good example. Hence a fundamental component of secure program design is to review developed code to ensure it adheres to known good programming practices, such as those distributed by the Computer Emergency Response Team (CERT) at `http://www.cert.org/secure-coding/scstandards.html`.

This assignment is intended as a warm-up for the coding you will do over the remainder of the semester. It requires you to read the code and find violations of good programming practices which you already know. These practices include the following.

- All return codes are checked and violations are handled in a way that adheres to the application's security design. (This typically requires a trade-off between security and user convenience. In your submissions over the semester, you should identify where a decision was made to favor user convenience over security and defend the decision.)
- Each variable is initialized before its value is used.
- Each function controls the value returned on all paths.
- There are no type conversion or overflow errors.
- User input is validated.
- All strings are null-terminated.
- There are no memory errors.
  - No invalid pointer is dereferenced.
  - All allocated memory is freed as soon as it is no longer in use and all memory is freed prior to program exit.
  - Allocated memory is not freed more than once.
  - It is not possible for the bounds of an array to be exceeded; library calls that will operate against unbounded buffers are not used.

2 Requirements

Attached is a poorly written application. You must rewrite the application so that it adheres to known good programming practices (and supports the intended functionality). There are two parts to the assignment.

2.1 Part 1

First you must examine the attached code and find the violations identified below. Note that you may have to read the manual pages to understand the operation of certain of the library routines. For each violation: (1) note the line number(s) that contains the violation and explain how these lines of code exhibit the violation and (2) explain (in writing) how the violation can be fixed. If you do not understand precisely how some portion of the code operates, use a C reference or develop your own test code to verify your understanding.

2.1.1 Violations

(a) Dynamic allocation for an array is in units that does not correspond to the array type.

(b) It is possible to write beyond the allocation for array `nameList` (can occur in seven different statements).

(c) It is possible to write beyond the allocation for array `fullName` (can occur in any of three different statements).

(d) Two variables are declared but never used.

(e) A variable is used before it is initialized.

(f) A pointer to dynamically allocated memory may be lost before the memory can be freed. (This means that it may not be possible to free the memory. It does not mean that the memory is not freed.)

(g) The value from the `typeList` subroutine may not always have been set to reflect an error.

(h) Return codes are not checked after each subroutine call

(i) There is a path through the code in which a pointer variable may not have been successfully initialized before it is used (that is, write to memory before successful allocation).

2.2 Part 2

Rewrite the code to fix the violations identified in Part 1 and any additional violations that you find. If you find additional violations, add them to the list in Part1 by including a line number(s), an explanation of the violation, and a description of your fix.

Also create a Makefile that compiles the application into a binary named `traverse` and that includes a `clean` directive to remove all but the source code.

3 Submission

The original C code is attached to this assignment. Your final code submission must also be written in C. An electronic copy of the original code will be made available `/classes/cs4471/projects/1.warmup/code/`.

Submit the following:

- A pdf report from Part 1.
- A copy of the source that is the final product from Part 2.
- A Makefile that supports the directives described above.
- A README that gives your name, the number of slip days (including zero) used for the assignment, and an overview of the contents of each file submitted.

For the second part of the project, use the `submit` command to submit a tar file named `warmup.tgz` that contains the source code, Makefile and README. When I execute the commands: `gtar zxf warmup.tgz; make;` against your submission, an executable named `traverse` should be created. The grader will execute `traverse` in order to grade your submission. He will assume that `traverse` expects the same two arguments (in the same order) as the code you were given.

4 Collaboration

Note that for this assignment, **no** collaboration is allowed. This means that the policy allowing empty-hands discussions does **not** apply to this assignment.

5 Grading

This project is worth five percent of the final course grade.