Homework 3

Fixing C code with Vulnerabilities

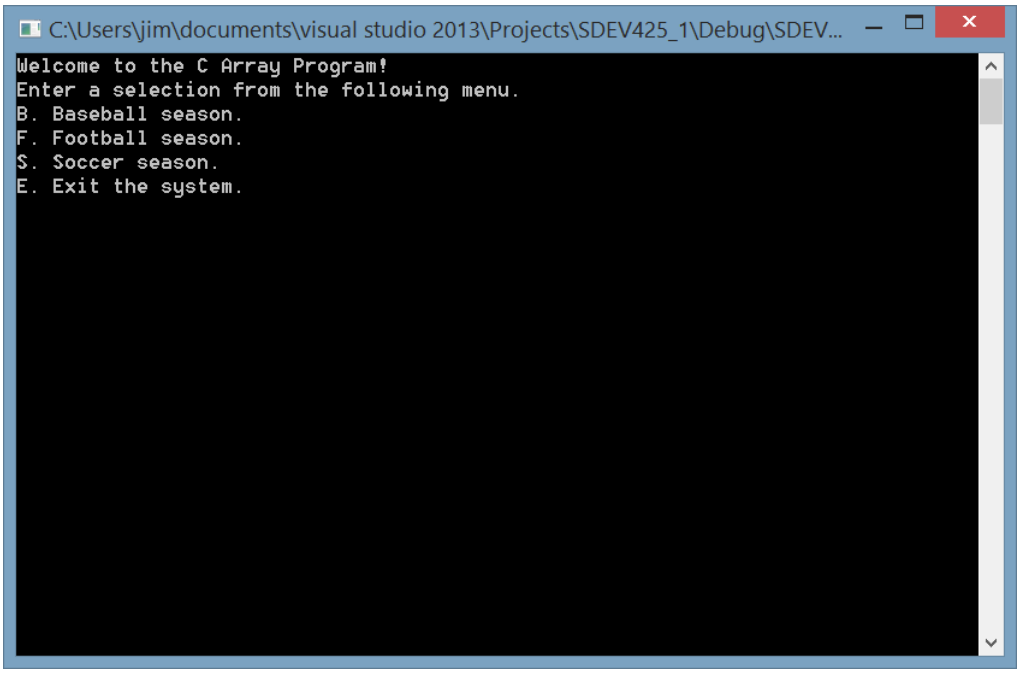
Overview

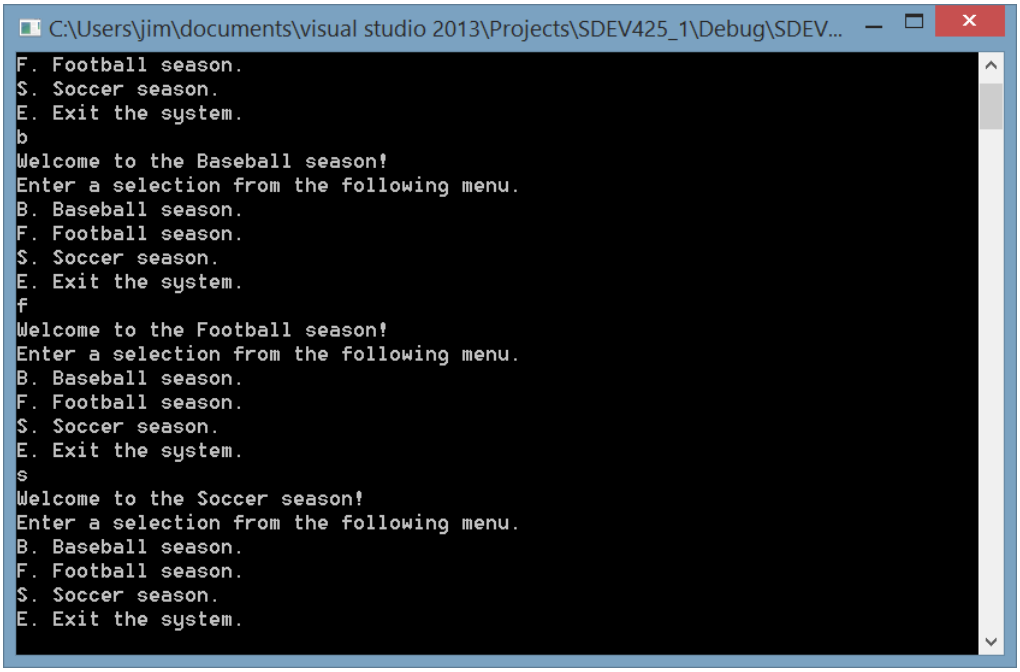
In this homework, you will modify an existing C code application that violates several C code rules and recommendations. Your task is to locate the issues, based on the readings for this course, identify the rule(s) or recommendation(s) being violated and then fix the code. You will discuss each issue in terms of why the issue may cause a security vulnerability, and how you specifically fixed the issue.

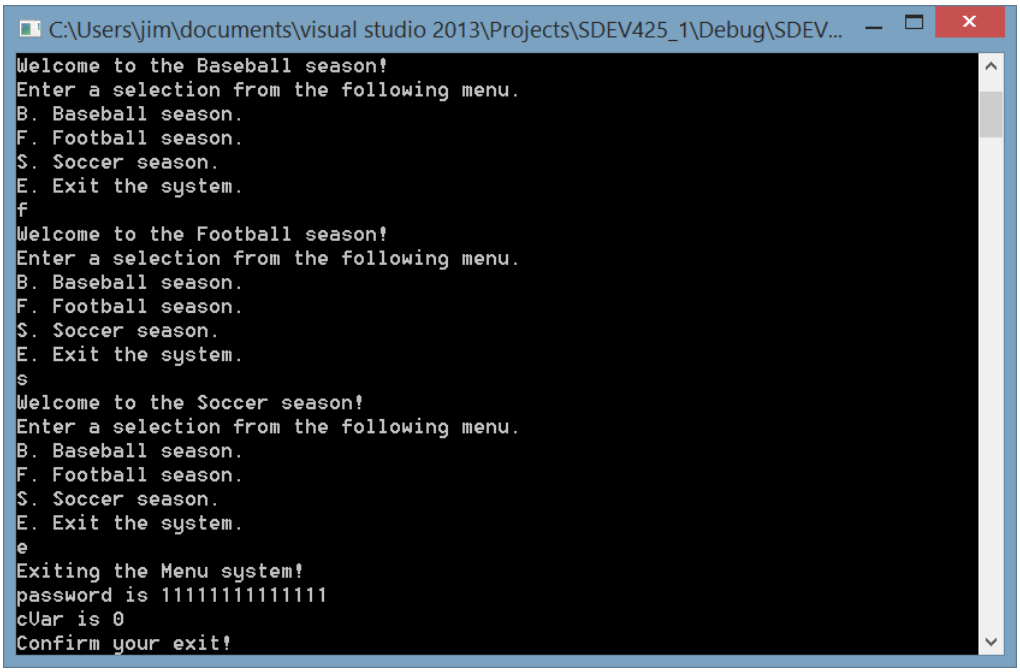
Assignment

Review and Understand the Sample C application.

The current code, developed by a junior developer, has several issues and is not functioning as expected. The desired functionality of the program is to allow a user to select from several choices on a menu. After the user selects the "Exit" option from the menu, the program will populate a password with '1's and then display the value of the password. The program also captures a character so the screen can stay paused for review before exiting. Below are screen shots for a successful program execution.







Unfortunately, not only are there security issues, the code you were provided doesn't work as expected.

For the first part of this exercise demonstrate your C developer environment is working properly. You can do this by running any of the sample C code applications.

Modify the C code in this example to make the desired functionality work properly. Demonstrate the code works properly through screen captures and describing what changes were made to fix the functionality issues.

Carefully, review the code and perform analysis as needed. Consider the following rules and recommendations and hints for items that you might want to review. Note, that some rules and recommendations listed below may not be found as issues in the code.

- STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator.
- MSC24-C. Do not use deprecated or obsolescent functions.
- FIO34-C. Distinguish between characters read from a file and EOF or WEOF.
- MSC17-C. Finish every set of statements associated with a case label with a break statement.
- MSC33-C. Do not pass invalid data to the asctime() function.
- MSC17-C. Finish every set of statements associated with a case label with a break statement.
- DCL20-C. Explicitly specify void when a function accepts no arguments.
- MEM30-C. Do not access freed memory.

You can use any C compiler you have access to including:

1. Windows C++ Express or Visual Studio

2. Mac X-Code C

3. Linux gcc

4. VM player with gcc (e.g. SDEV 300 Virtual machine)

Be sure you have a C environment where you can compile. Also review those code tutorial links provided in the classroom. Post a note, or contact your professor if you are having significant difficulties compiling a C program.

Once you have your environment working, reviewed and analyzed the code, and determined the rules and recommendations that are violated, you should fix the code. Be sure to document each issue by aligning the rule or recommendation and explain exactly how you fixed the issue.

Hints:

a. Make sure your C coding environment is working first. Those C tutorials will help you to test your environment.
b. Be very careful with the pointers and memory limits of the arrays. Most modern compilers attempt to protect your system resources, but you could potentially produce access violations that could lock your system up. Take your time and review the memory bounds for all of your arrays before you start making code changes
c. Start on this early. This will take you longer than you think.

Deliverables

Provide your fixed C source code along with a PDF document describing how you addressed each issue. For example, you should list the C Cert rule or recommendation for each issue and show and

describe the code that addresses the issue. You should also provide screen shots and descriptions of the successful execution of the code.

Be sure your PDF document is neat, well-organized and is well-written with minimal spelling and grammar errors. All references used should be included in your document.

Grading rubric:

Attribute	**Meets**	**Does not meet**
Sample C code application	**10 points** Demonstrate your C developer environment is working properly. (5 points) Modify the C code to make the desired functionality work properly. Demonstrate the code works properly. (5 points)	**0 points** Does not demonstrate your C developer environment is working properly. Does not modify the C code to make the desired functionality work properly. Does not demonstrate the code works properly.
C code rules and recommendations	**70 points** Applies STR31-C, if needed, as needed to guarantee that storage for strings has sufficient space for character data and the null terminator. (10 points) Applies MSC24-C, if needed, to not use deprecated or obsolescent functions. (10 points) Applies FIO34-C, if needed, to distinguish between characters read from a file and EOF or WEOF. (10 points) Applies MSC17-C, if needed, to finish every set of statements associated with a case label with a break statement. (10 points) Applies MSC33-C, if needed, to not pass invalid data to the asctime() function.(5 points) Applies MSC17-C, if needed, to finish every set of statements associated with a case label with a break statement. (5 points)	**0 points** Does not apply STR31-C, if needed, as needed to guarantee that storage for strings has sufficient space for character data and the null terminator. Does not apply, if needed, to not use deprecated or obsolescent functions. Does not apply, if needed, to distinguish between characters read from a file and EOF or WEOF. Does not apply, if needed, to finish every set of statements associated with a case label with a break statement. Does not apply, if needed, to not pass invalid data to the asctime() function. Does not apply, if needed, to finish every set of statements associated with a case label with a break statement. Does not apply DCL20-C, if needed, to explicitly specify void when a function accepts no arguments. Does not apply MEM30-C, if needed, to not access freed memory.

	Applies DCL20-C, if needed, to explicitly specify void when a function accepts no arguments. (10 points) Applies MEM30-C, if needed, to not access freed memory. (10 points)	
Documentation and Submission	**20 points** Provides all C source code including "fixed" code. (5 points) Provides screen shots and descriptions of the successful executing the code and the resultant output as applied to each security control. (5 points) Document is neat, well-organized and is well-written with minimal spelling and grammar errors. (5points) All references used should be included in your document. (5 points)	**0 points** Does not provide all Java source code including "fixed" code. Does not provide screen shots and descriptions of the successful executing the code and the resultant output as applied to each security control. Document is not neat, well-organized and is not well-written with minimal spelling and grammar errors. All references used were not included in your document.