Cyber Range Lab Assignment 8

C++ Code Security

Jon Mahoney

SRA 440W

# General Context

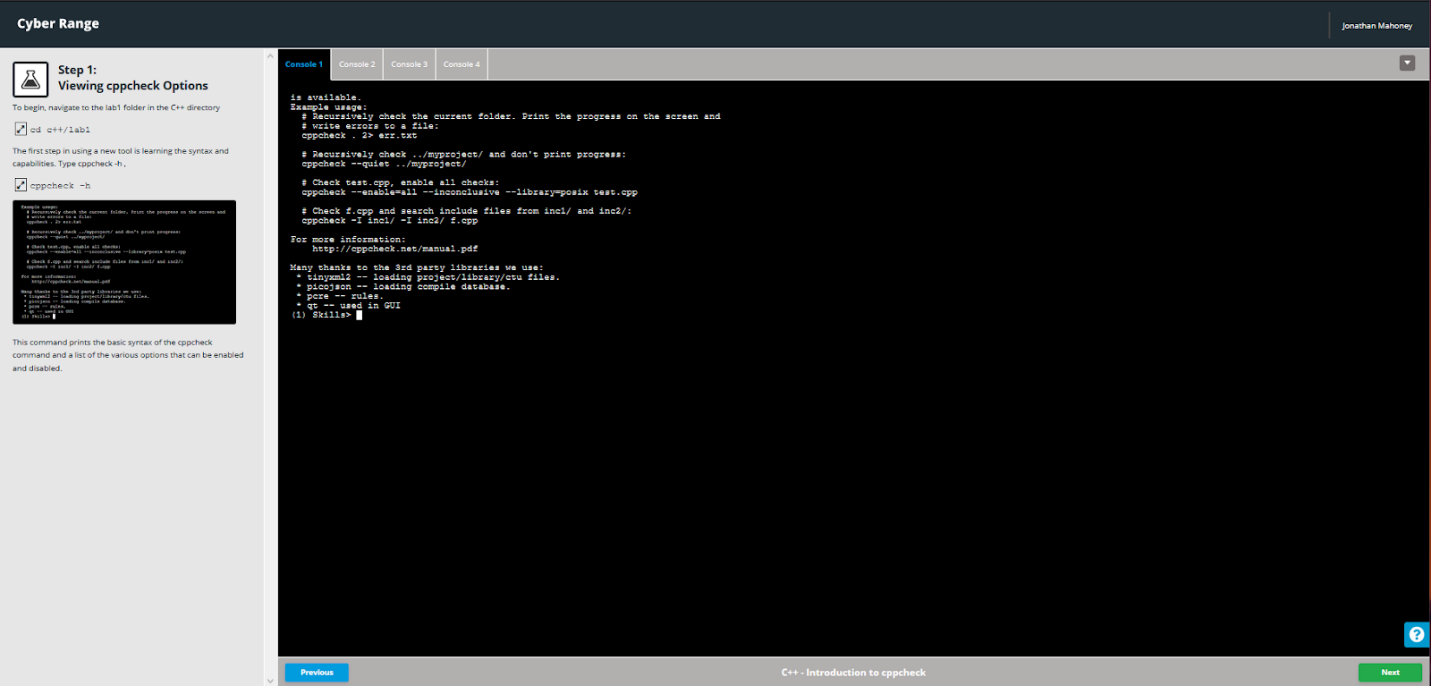
Lab assignment 8 contains four labs which include exercises on vulnerabilities in C++ and how to address them. The first lab of the assignment is Introduction to cppcheck which introduces the static analysis tool cppcheck. In the lab, cppcheck is used to test C++ files for vulnerabilities and the instructions describe the best practices in doing so. For instance, instead of checking an entire directory for vulnerabilities, it is often better to specify a range of files to be tested. One of the vulnerabilities found in this section was a segmentation fault resulting from a NULL pointer, or a pointer referring to memory addresses it does not have access to. The next lab, Integer Overflows and Underflows, describes the issues that can arise through careless typecasting of variables. Unsigned integers are a data type that cannot be negative whereas a signed integer can be negative. In other words, the lab describes the issues that can arise due to improper casting of integers that should be unsigned, to signed. The next lab, Type Management, involves a similar topic to the previous lab which is type casting. However, the lab includes other variable types besides integers, such as the floating-point data type which is primarily used in functions that rely on numbers with decimals. Lastly, the lab Poor Error Handling covers some common mistakes in C++ programs such as the use of NULL pointers, lack of error handling, and bad switch statements. Additionally, the lab covers some of the best practices to avoid these issues through error handling. As a general rule of thumb, developers should assume users will enter values their program did not ask for and should prepare error handling for every possible case.

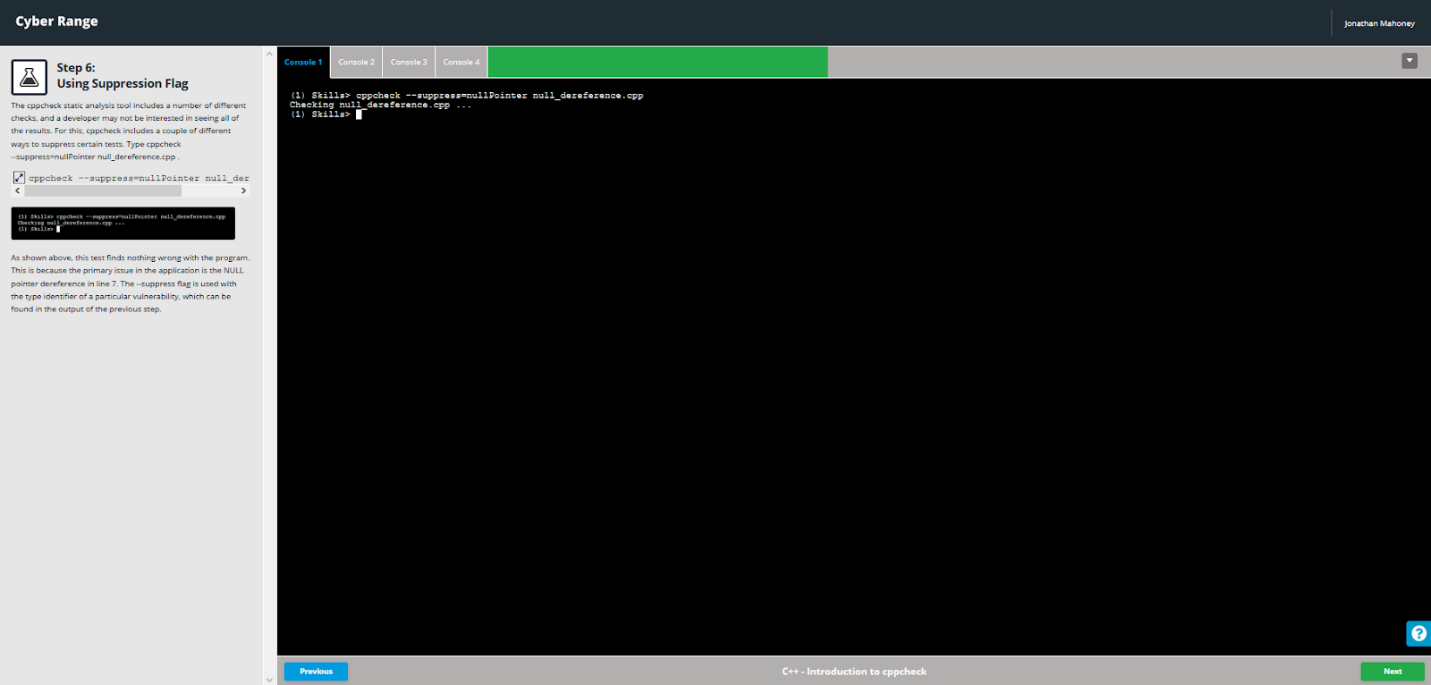
# Solution

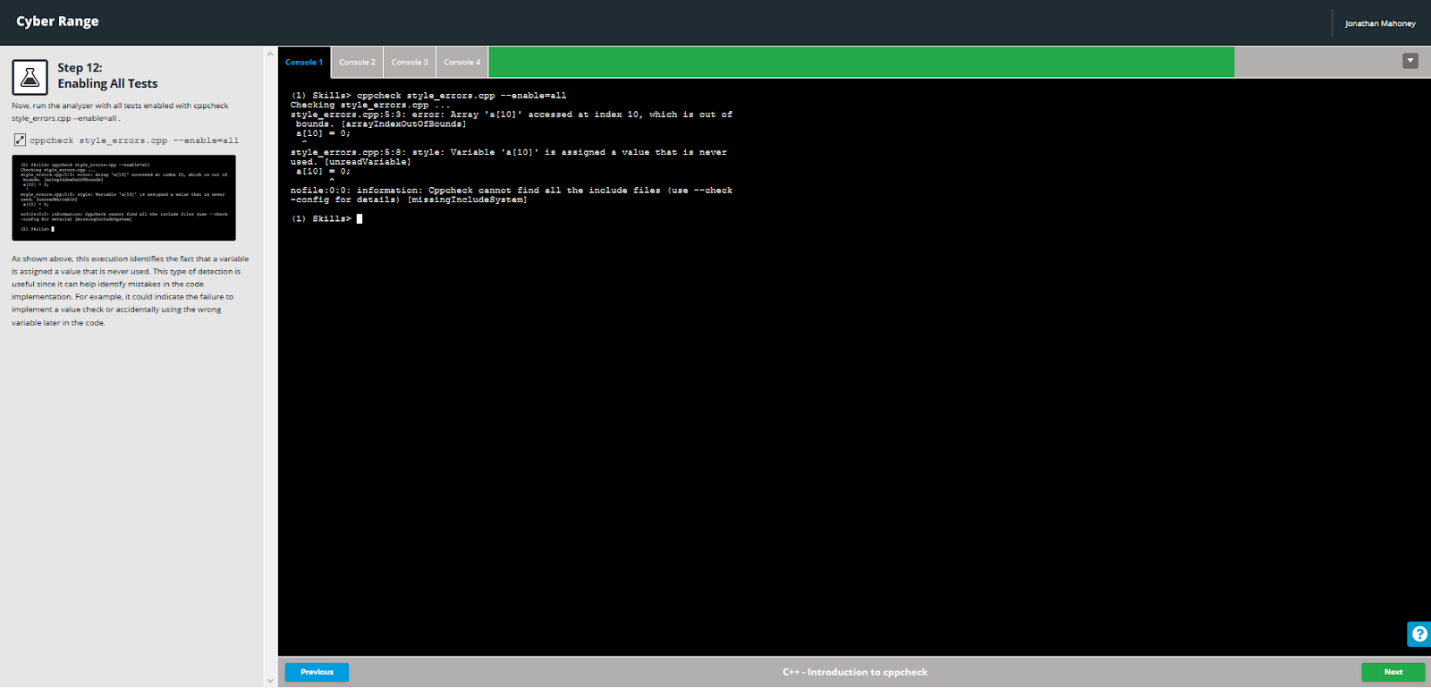
This was a lab that I especially found familiar due to the fact that I had taken two semesters of C++ classes, so I could tell what was going on in the text editors. Specifically, CMPSCI 101 provided me with most of the information I needed for the lab. As an example, a topic that was often discussed in the course was error handling and it was actually a criteria our programs would be graded on. Additionally, this lab followed a similar format to the previous Python lab where the lab walks through a vulnerability and the following section describes the best practices to avoid the vulnerability. Personally speaking, I have found that it is best to limit user input to reduce the likelihood of improperly handling errors. For instance, if you have someone enter their state of residence into an online form, you should use a drop-down box for selection as opposed to a text entry for typing as some people may misspell their state. If you have the user select their state from a list instead of typing it out, you eliminate the need to handle cases of misspelled states.

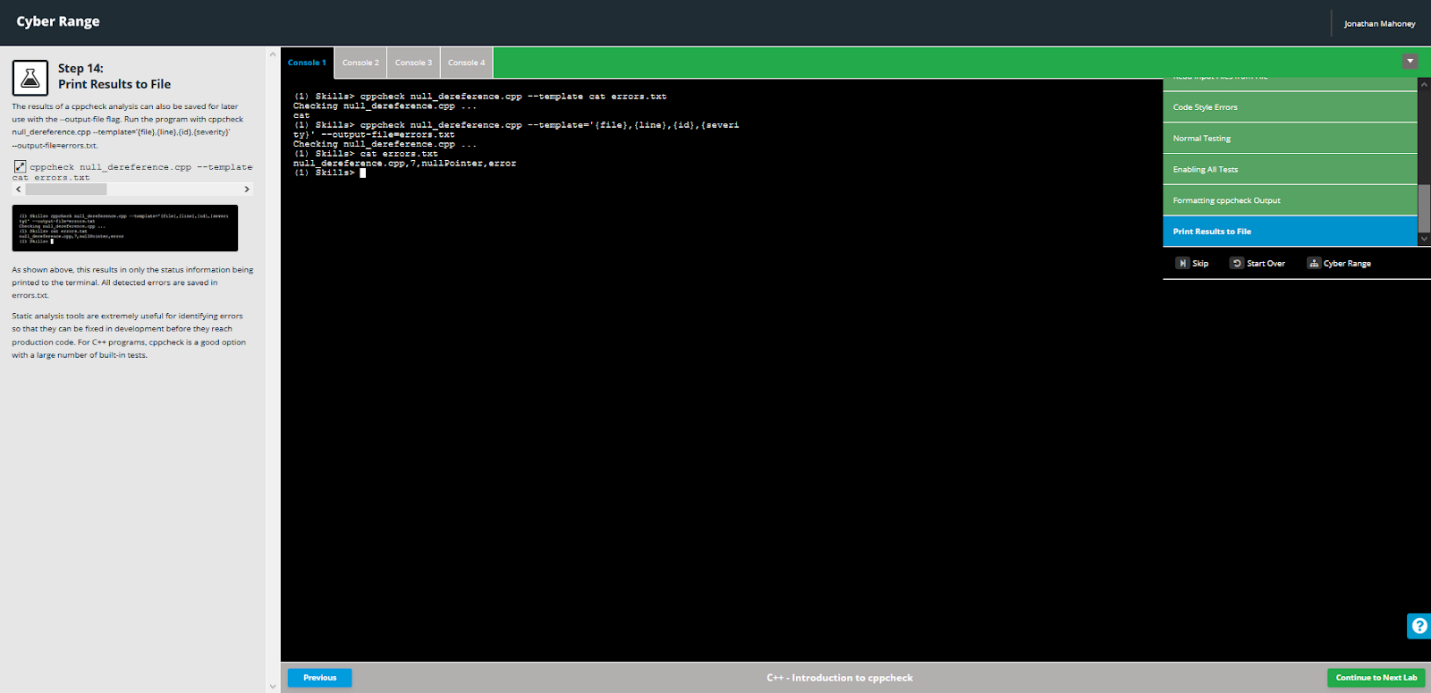
# Screenshots

## Introduction to cppcheck

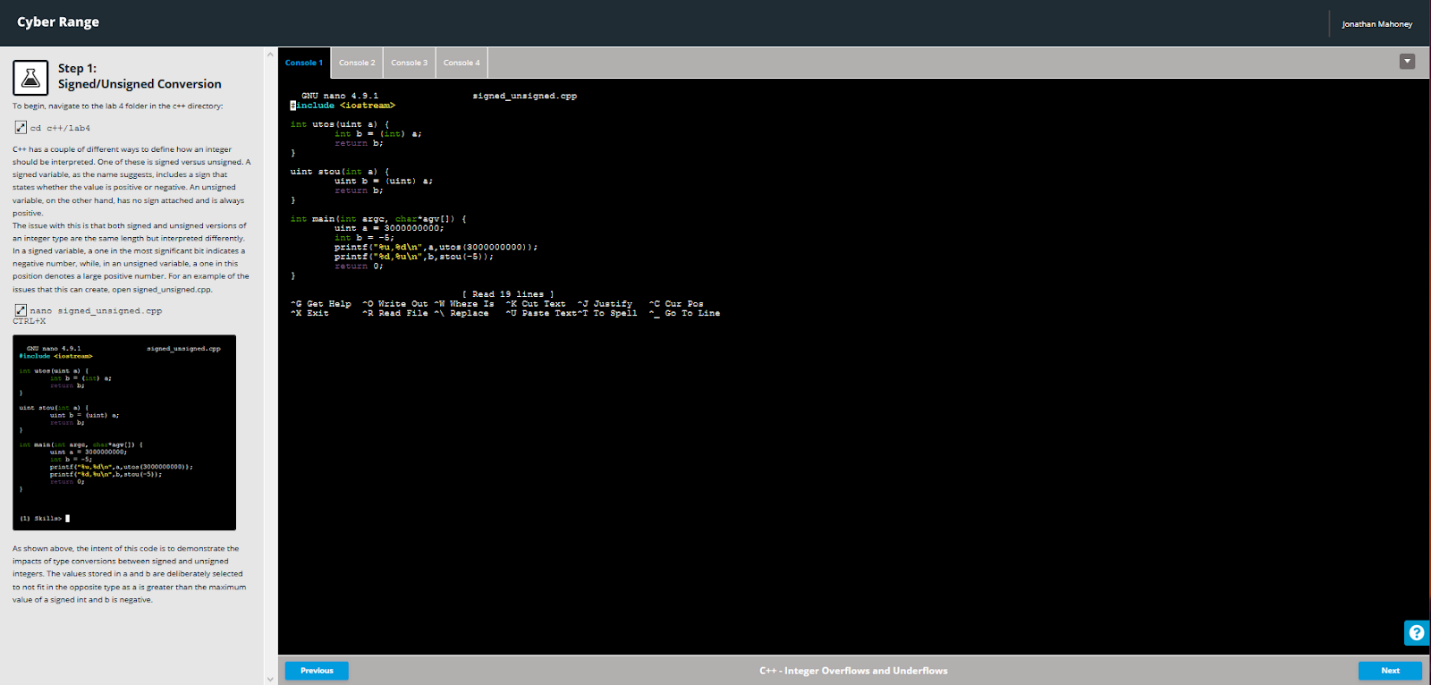


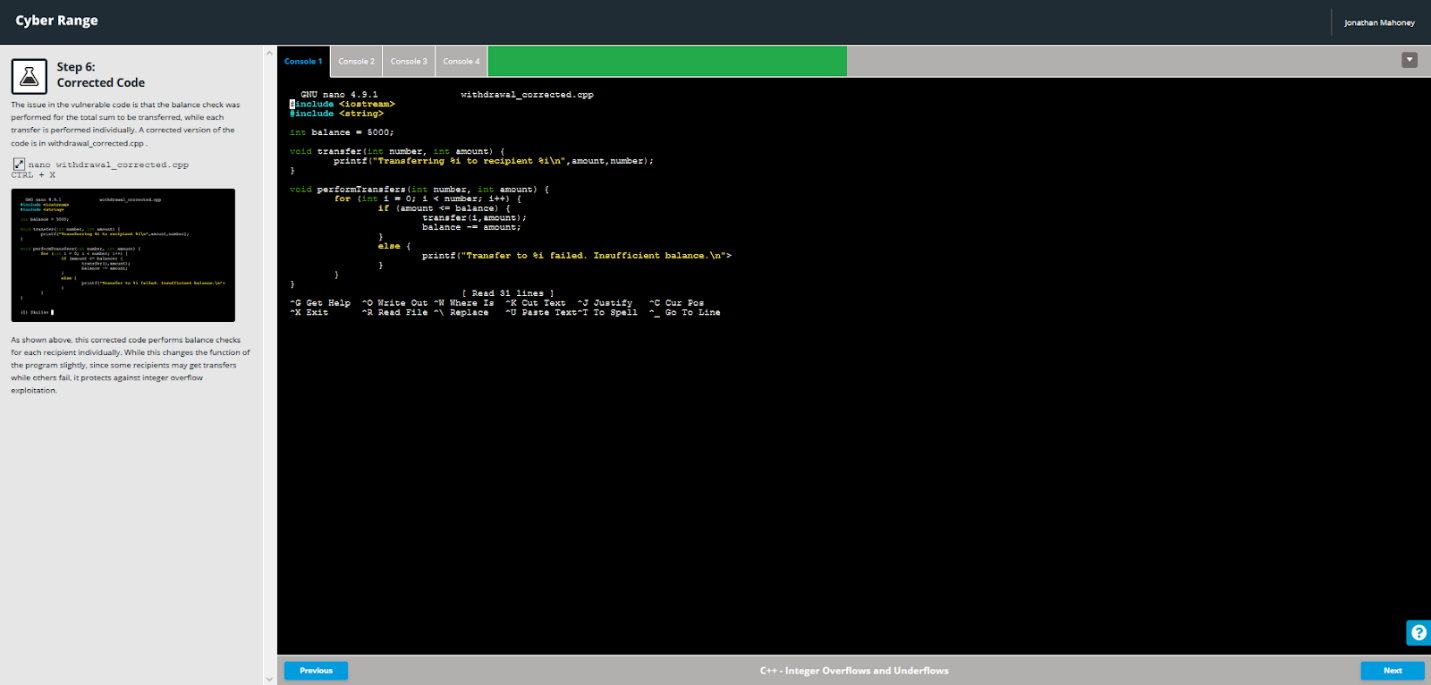


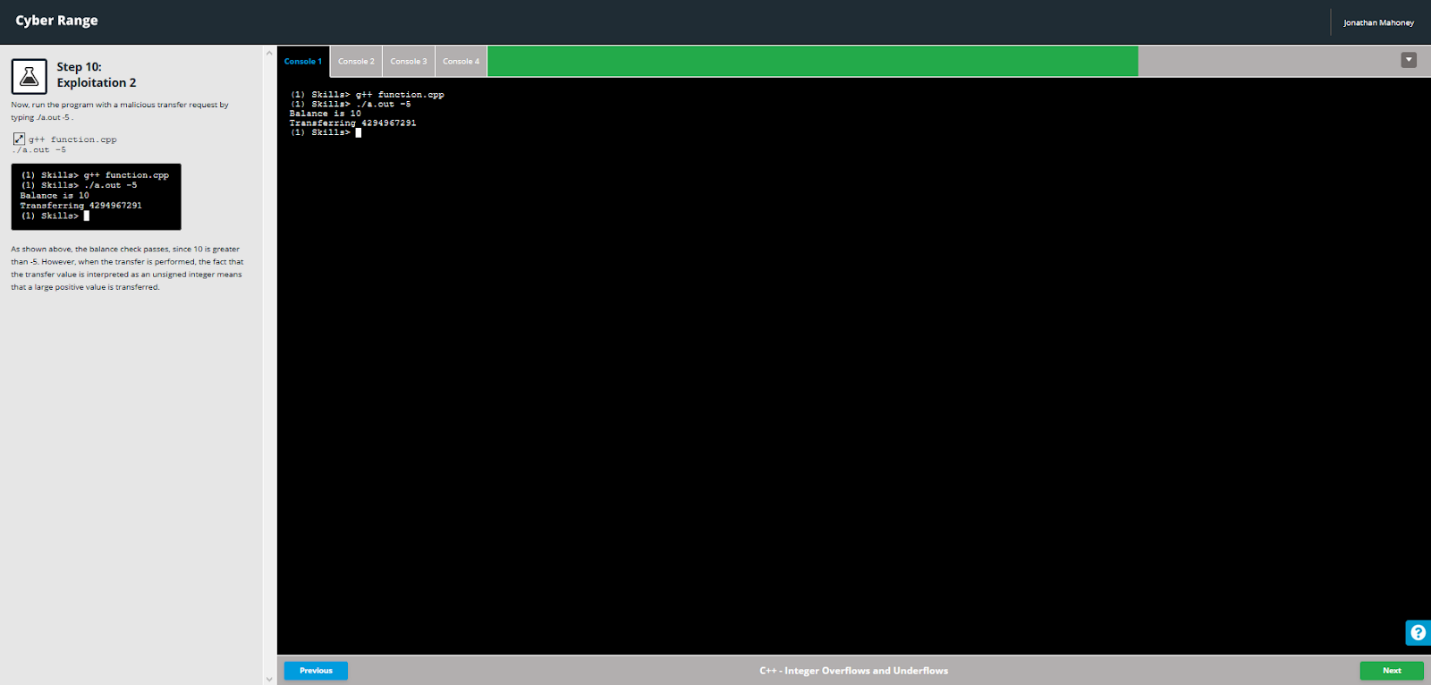


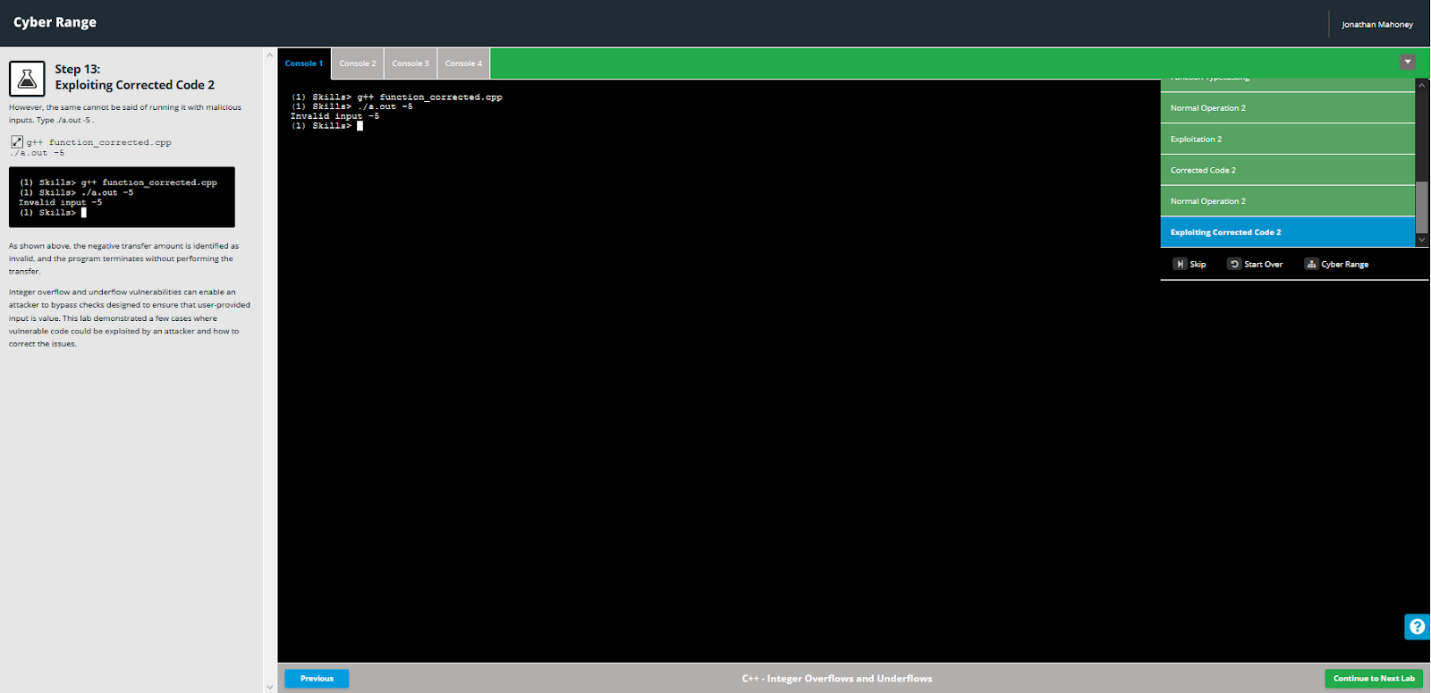


## Integer Overflows and Underflows

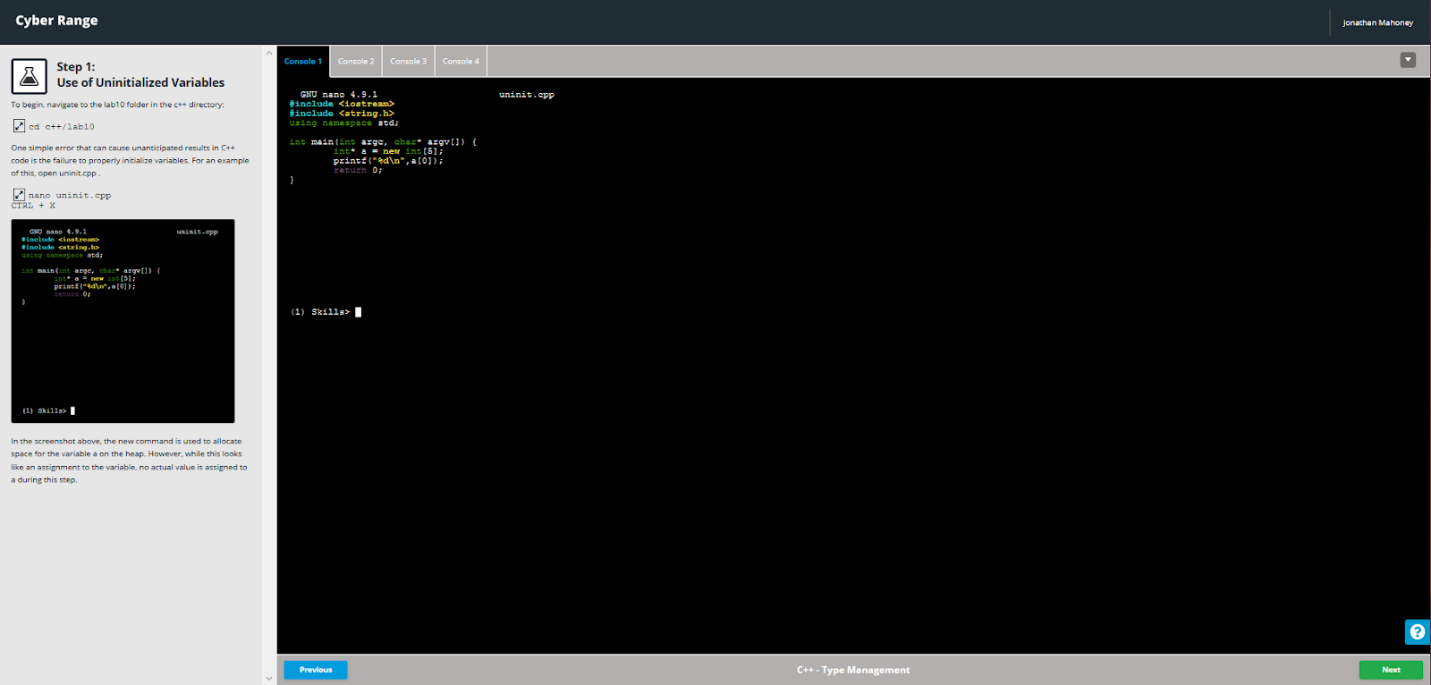


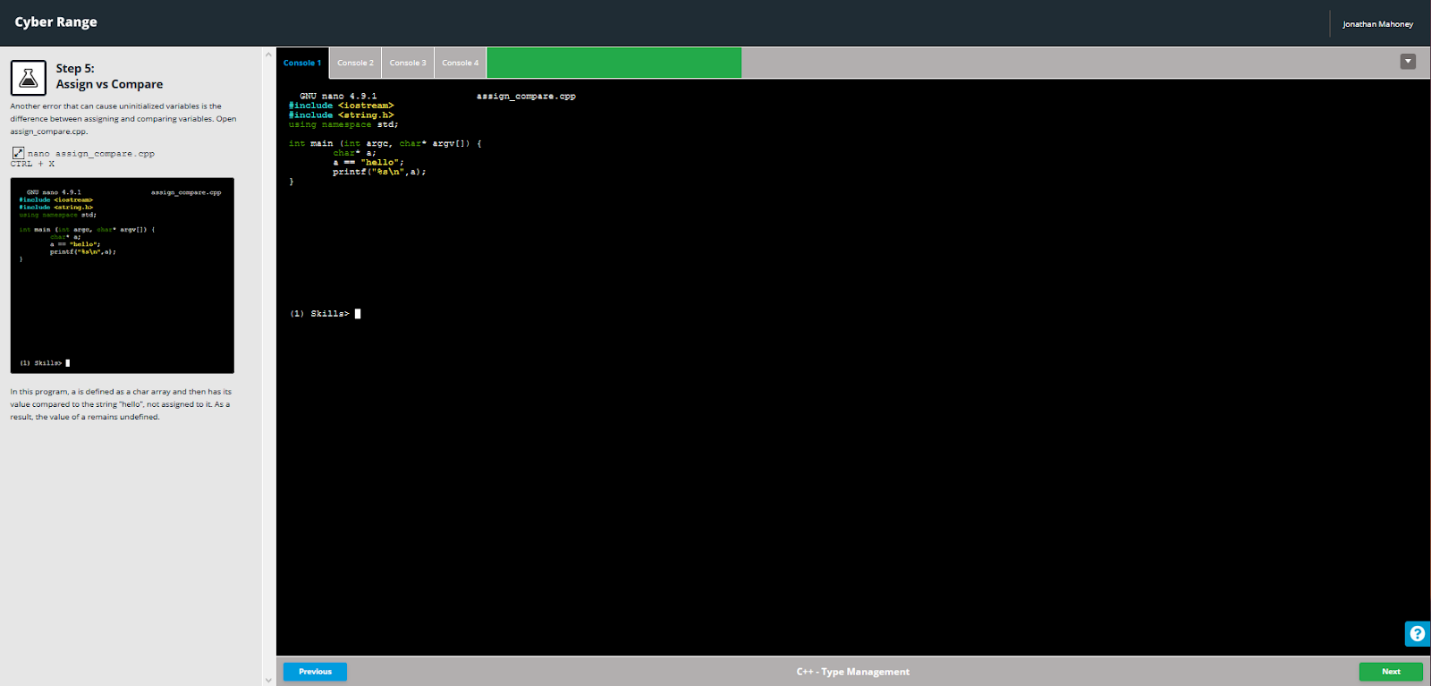


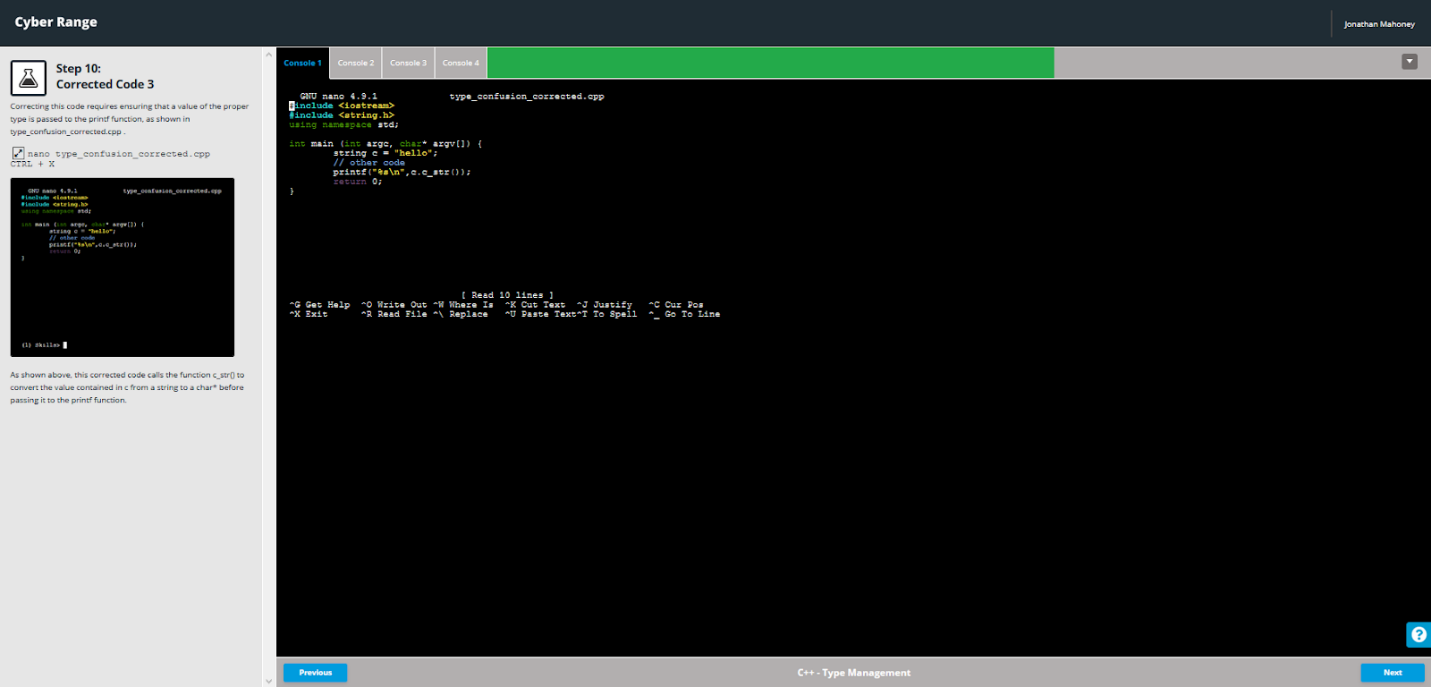


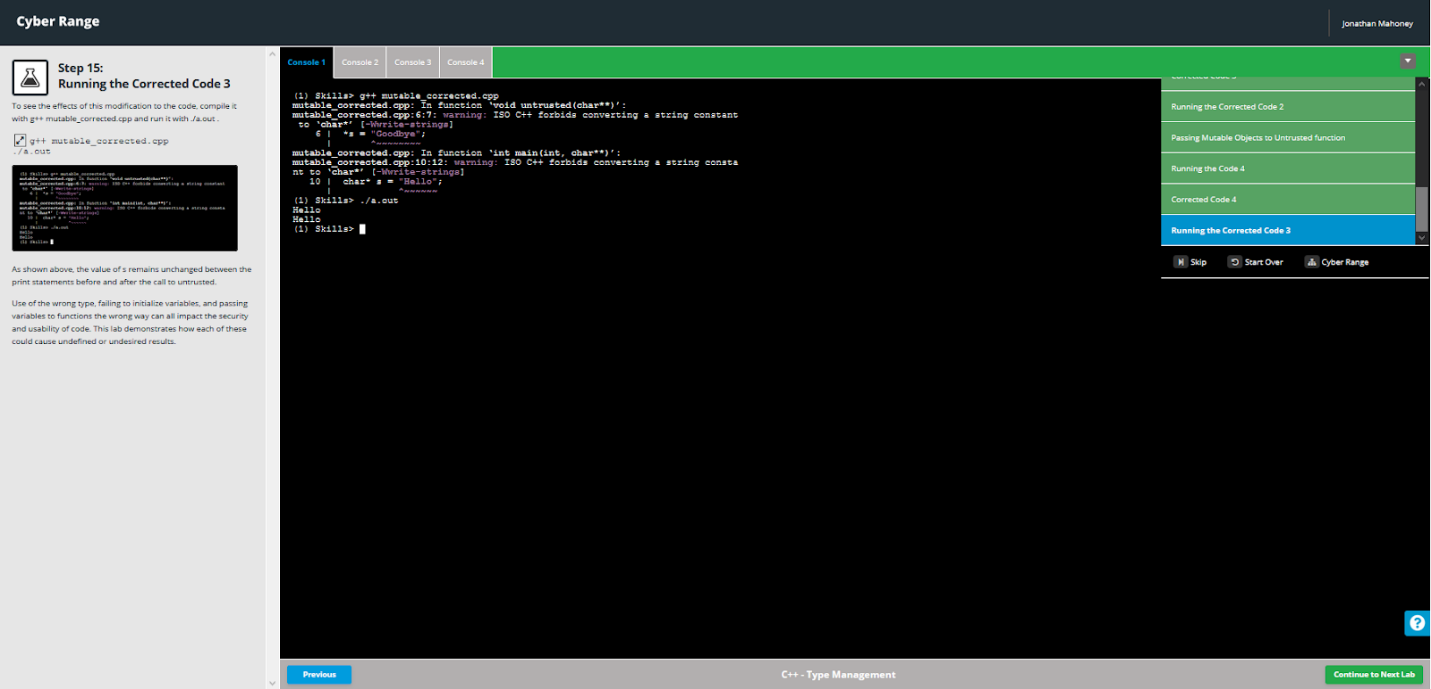


## Type Management









## Poor Error Handling

