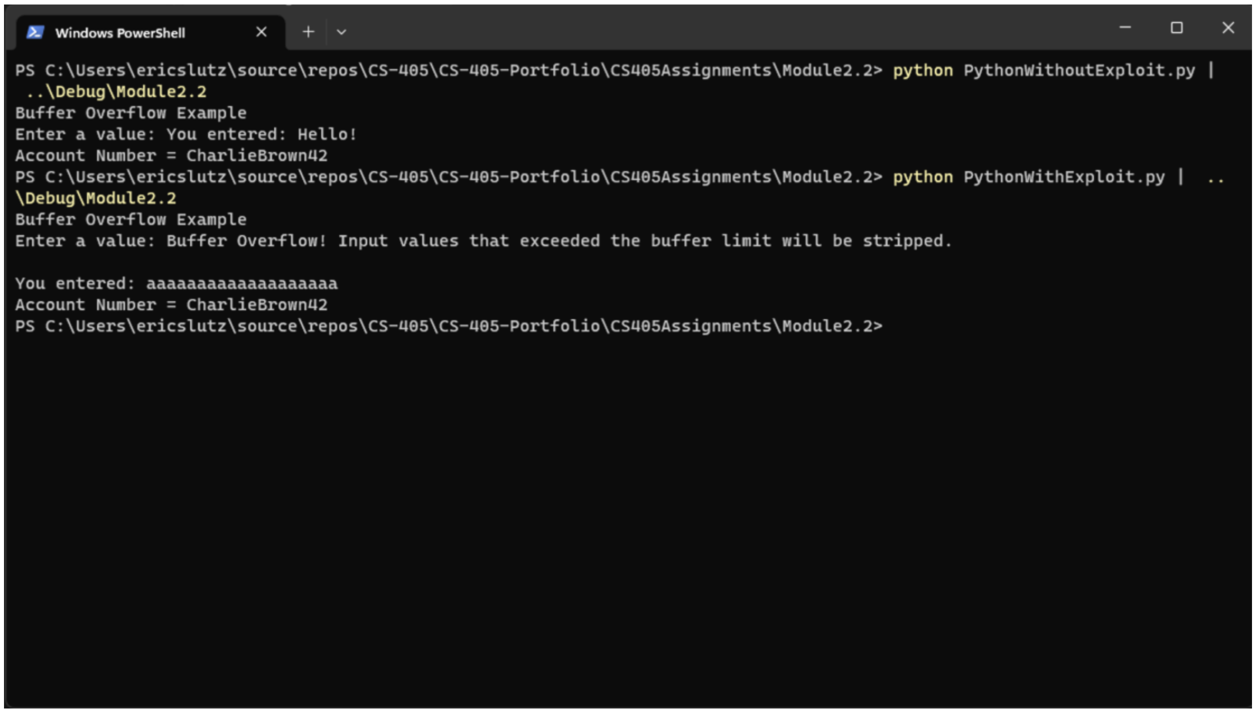
CS-405 Secure Coding – Module 2 Assignment 2

Eric Slutz

Southern New Hampshire University

Module Two Assignment Two

Buffer overflow test screenshot



```
PS C:\Users\ericslutz\source\repos\CS-405\CS-405-Portfolio\CS405Assignments\Module2.2> python PythonWithoutExploit.py |
 ..\Debug\Module2.2
Buffer Overflow Example
Enter a value: You entered: Hello!
Account Number = CharlieBrown42
PS C:\Users\ericslutz\source\repos\CS-405\CS-405-Portfolio\CS405Assignments\Module2.2> python PythonWithExploit.py |  ..
\Debug\Module2.2
Buffer Overflow Example
Enter a value: Buffer Overflow! Input values that exceeded the buffer limit will be stripped.

You entered: aaaaaaaaaaaaaaaaaaa
Account Number = CharlieBrown42
PS C:\Users\ericslutz\source\repos\CS-405\CS-405-Portfolio\CS405Assignments\Module2.2>
```

Summary of buffer overflow detection process

For this assignment, you were given a char array that could potentially overflow when given input. The best way to avoid this buffer overflow issue is to set user_input as a string and not as a char array as strings do not have the same overflow issues. However, I felt that defeats the purpose of this assignment, so I looked for another method. Using std::cin.getline() you are able to set the variable you want the input assigned to as well as how much input you want to receive (cin.getline(user_input, MAX_INPUT_LENGTH)). Now that this was handing potential buffer overflow, the next step was to alert the user when an overflow happens. The first step is to check if the input exceeded 20 characters through the use of conditionals such as !cin. If there are no issues, then the user input and the unmodified account number are displayed. If there was a potential buffer overflow, then a waring is displayed, and the cin errors and buffers are cleared.