CS165 Project 3 Documentation

Team Members:
Arnel Soriano - asori047
Edwin Lopez - elope083

Part 1: The objective is to leverage the buffer overflow vulnerability present in the handling of the answer field within the string_q structure, aiming to overwrite a function pointer and redirect execution to a function that prints "hacked!" to the terminal.

Methodology and thought process:

```
#Identified that string_q is susceptible to overflow in the answer via strcpy.
#Determined exact size needed for overflow by testing with different sizes of input.
#Found that 55 'A's causes overflow, leading to heap corruption.
#Utilized GDB to monitor the program's behavior and inspect the memory layout before and after the overflow.
break add_question
run commands.txt output.txt
#Before overflow: Inspected sq to understand its initial state.
p sq
x/20wx sq
#Stepped through the addition of a question to observe memory changes.
next
#After overflow: Examined sq again to see the effects of the overflow.
p sq
x/20wx sq
#The memory inspection revealed 'A's filling and exceeding the answer buffer, indicating successful overflow.
#Aimed to refine the payload to control the overflow precisely, targeting the overwrite of a function pointer within sq.

#Adjustments made to payload to target function pointer accurately, considering structure layout and offset calculations.
#Testing different payloads to observe changes in program behavior and confirm successful overwrite of the function pointer.
#Encountered heap corruption errors ("malloc(): corrupted top size") upon executing the payload, indicating an overflow impact on the heap.
```

To Finish:

- Further refine the payload to ensure precise control over what gets overwritten, aiming for a function pointer within sq.

- Utilize GDB to step through the program more granularly, inspecting memory just before and after critical operations to better understand the overflow's impact.
- Plan to adjust the overflow size and content based on a deeper understanding of the string_q structure's layout and adjacent memory, ensuring the payload achieves the desired effect without unintended side effects.

Part 2: Demonstrate a buffer overread vulnerability to leak sensitive information, specifically the answer value of an integer question object, without causing a crash or overtly altering the program's flow.

- Initially, focus on identifying where the program handles integer question-answer pairs, particularly how the answers are stored and accessed.
- Investigate the program's behavior when processing overly long inputs for integer-type questions, looking for opportunities where memory beyond the intended buffer might be read.
- Use GDB to inspect the memory layout of the integer question-answer structures to understand how they are organized and to identify potential points where an overread could occur.
- Create a specific input or series of inputs that trigger the buffer overread, aiming to leak the answer to an integer question.

```
Program is crashing due to heap corruption, and stepping through library
code hasn't provided useful insight into the source of the corruption
```

To Do

- Debug step by step and analyze memory allocation and usage

Part 3: Leverage type confusion vulnerabilities to alter the execution flow of the cs165-p3 program, specifically by overwriting a function pointer within a string_q structure to point to the printf function, demonstrating arbitrary code execution or, in this controlled scenario, unintended behavior.

Current Progress and Findings:

- Successfully induced a buffer overflow affecting the string_q structure's function pointer, evidenced by memory inspection before and after the overflow operation.
- Encountered challenges in precisely controlling the overflow to avoid heap corruption, indicated by program crashes following the execution of manipulated payloads.
- Have yet to achieve the desired redirection of execution flow to the printf function, necessitating further payload refinement and possibly deeper investigation into the program's memory management and structure handling.

To Do: Detailed memory inspection