Project 1 - Part 1: Overflowing the Stack

Goals of the Project:

- Understand and explain how virtual memory is laid out into different regions
- Understand and explain how the stack and heap work
- Understand and explain the basic stack concepts and how it controls program execution
- Understand and explain the concepts of buffer overflow
- Explain how a buffer overflow occurs in detail and what effects it has on the heap

Supplemental Readings:

- *Complete Virtual Memory Systems*
- *Memory API*
- *Address Spaces*

The final deliverable:

The submitted paper must adhere to the following format: (**-10** for violation of **ANY** of the following)

- Written answers to each individual question are limited to a **maximum** of 150 words
- Font: Times New Roman, Size 11 (or JDF default).
- Spacing: Single Spaced with standard 1" margins
- Standard page size required. (8.5" x 11")
- Heading: students' gt_user_id used in header with page number
- Final deliverable **MUST** be in **pdf** format. File name format: **`gt_user_id_Project1.pdf`**
  - Example: **`gburdell27_Project1.pdf`**
  - We know canvas inserts a -1 after a submission. This is fine.
- You are allowed to submit the paper in JDF (Joyner Document Format)

Important: Please do not just copy this document and insert your answers. Format your submission in a way that is easy to read and easy to grade. We appreciate this. (**-10** if this is not adhered to)

Information:

- ***Plagiarism will not be tolerated!*** For information: GaTech Academic Honor Code and the Syllabus.
- You **MUST** use the latest version of VirtualBox, downloaded from: VirtualBox
- GDB command cheat sheet: Cheat Sheet
- GDB Screenshots **DO NOT** count as an explanation. You can use them as a tool, but you are not allowed to just paste a screenshot. Explain in detail what that screenshot is showing. Highlight, circle, underline, etc…
- Refer to the "Deductions" section to ensure you do not lose unnecessary points.

Project Tasks (75 points):

I. Understanding Foundational Concepts - (22 points)

1. What is a pointer? **(1 point)**
2. How is memory managed differently between "C" and "Java"? How is it allocated and how is it deallocated? **(2 points)**
3. What is the difference between "pass by value" and "pass by reference"? **(1 point)**
4. Why would it be smart to use fixed-width integer types? **(1 point)**
5. What does it mean when a function or variable is static in the "C" language? **(2 points)**
6. What is the difference between "compile time" and "run time"? Give an example of something that may be allocated at compile time, and describe some mechanisms for allocating memory at runtime. **(3 points)**
7. What is the difference between an assembler, compiler, and an interpreter? **(3 points)**
8. Is C a compiled language? Is Python? If either answer is no, how is the language processed so code can be run? **(2 points)**
9. What is the GNU compiler? How do you invoke the GNU Compiler? **(2 points)**
10. Explain what GDB is, then explain how you would use it to do the following: **(5 points)**
    a. View the processor registers
    b. Set a breakpoint in code
    c. Find the address of an OS Function
    d. Inspect a memory location

II. Understanding Components - (10 points)

Important: Answer the following questions in relation to the 32-bit Linux VM that is being used, unless otherwise noted. Research each of the following and answer all questions regarding them:

1. Address space layout randomization (ASLR) **(2 points)**
    - What is ASLR? How does it affect the stack?
    - How can ASLR be bypassed without turning it off?
2. Stack Canary **(2 points)**
    - What is a Stack Canary? How does it affect the stack?
    - Are Stack Canaries vulnerable and if so, how?
3. How does a Stack Buffer Overflow (with no protections enabled) affect the stack? **(1 point)**
4. Which registers are stack specific and what is the purpose of each (x86 specific for this question) **(2 points)**
5. There exists the concept of word-alignment on the stack. Describe what this is and how it affects the stack in a 32-bit OS that utilizes it. **(3 points)**
    - Based on this, how would the following lines of code be allocated on the stack?

```
char buf1[5]; // Line 1
char buf2[8]; // Line 2
```

III. Compiler Flag Options - (4 points)

Research each of the following and give an explanation to how each specifically affects the program/binary. *Important: This is referring to the **gcc** compiler.* **(1 point each)**

-O0 (This is the letter **O**, then the numeral **0**)
-g
-fno-stack-protector
-z execstack

IV. Buffer and Heap Understanding - (14 points)

All of the following questions refer to the address space of a 32 bit Linux distribution OS.

1. When is memory allocated and de-allocated on the stack in C? **(2 points)**
2. How is data stored and organized on the stack? **(2 points)**
3. Where in memory is the stack located? **(1 point)**
4. When is memory allocated and de-allocated on the heap in C? **(2 points)**
5. Explain how data is stored and organized on the heap using chunks. **(3 points)**
6. Where in memory is the heap located? **(1 point)**
7. How is the stack used to control program flow when a function is called? **(3 points)**

V. Stack Overflow - (25 points)

The VM is located: VM Download Link and the code provided below is in a file **`overflow.c`** which can be downloaded here. Figure out how to compile and run the program. Be sure you understand how the overflow works.

Hint: You may need to use the research you did above to get this program to execute correctly.

```c
#include <stdio.h>
#include <string.h>

int main (int argc, char *argv [])
{
    int access = 0;
    char str[8];

    printf(“Please enter a password: “);
    scanf("%s", str);

    if (strncmp (str, "password", 8) == 0)
        access = 1;
    if (access > 0)
        printf("Access Granted!\n");
}
```

Important: Include a screenshot of your terminal exploiting overflow.c including the gcc compilation (outside of GDB) in the write up. (If not, **-10 points**)
Important: The output “***stack smashing detected***” should not happen! (**-5 points** if this happens)

Stack Overflow Understanding (7 points)

1. Without entering “password”, what could you enter in order to display “Access Granted!”?**(2 points)**
2. Why is this vulnerability possible and how does it work? **(2 points)**
3. How could you change the code to make this exploit not possible?**(2 points)**
4. Would a similar program have the same vulnerability in a strongly typed language? Why or why not? **(1 point)**

Stack Overflow Diagram (18 points)

Create a diagram that shows what the stack would look like before and after the given input "OverflowDone" for ***overflow.c***. This diagram has to be 100% created by you. You are ***not*** allowed to copy/paste anything from anywhere else. Your diagram **cannot** be hand drawn. Your diagram must include:

- The order of parameters, return address, saved registers, and local variable(s). **(8 points)**
- The sizes in bytes. **(4 points)**
- The overflow direction in the stack. **(2 points)**
- Size of the overflowing buffer. **(4 points)**

Important: The sample input entered and shown in your diagram must be "OverflowDone".
(5 point deduction for any other input)
Important: No hand drawn diagrams.
(full 18 point deduction for missing diagram or hand drawn diagram)

Deductions Summary:

These are cumulative but will not result in a negative final grade:

- Any written answer exceeds 150 words (-10 points)
- Document not formatted as follows:
    - Font: Times New Roman, size 11 (-10 points)
    - 8.5 x 11 inch page size (-10 points)
    - single-spaced and 1 inch margin (-10 points)
    - PDF format (-10 points)
- Students' gt_user_id not used in header with page number (-10 points)
- Final document copy of original with answers entered (-10 points)
- Source citations do not adhere to IEEE format (-10 points)
- Hand drawn diagram used for Part V (-18 points)
- Input other than "OverflowDone" used in Part V (-5 points)