Project 1 - Part 1 :
Stack Overflow Case Study

The goal of Project 1 is to simulate how an attacker can take advantage of possible vulnerabilities in memory to hijack program control flow and gain higher system privileges. Project 1 is broken into two parts: Part 1 - Quiz (20%) and Part 2 - Code (80%)

Learning Goals of this Project:

- How virtual memory is laid out into different regions
- How the stack and heap work
- Basic stack concepts and how it controls program execution
- Concepts of buffer overflow
- Countermeasures to buffer overflow attacks
- GNU, GDB, and GCC basics

Deliverables:

- Task I is a quiz format to be completed on canvas named **"Project 1 - Part 1: Stack Overflow Case Study"**. It is **NOT "Quiz 1"**. Be very careful about which quiz you are taking. We will not give you another chance on a quiz if you take the wrong one due to not reading these directions carefully. **This is the only deliverable for Project 1 Part 1.**
- Tasks II & III are open discussion topics in edStem where students can collaborate. There is no deliverable for these tasks and they do not count toward your grade. Learn and enjoy!

Quiz Details:

- You may attempt the quiz any number of times during the window in which it is open.
- Feedback will not be given until after the quiz has closed.
- There is no time limit.
- Only your latest attempt will be graded.
- Use the information from Task I of this writeup and your research to complete it.

Additional Information:

- Be sure to not submit late as there is a late penalty specified in the syllabus that will be strictly enforced.
    - Remember to submit your quiz before the deadline. **If you start another attempt and leave it open, it will autosubmit at the until date** and be late (and possibly receive a 0 if you did not answer the questions). **We will not correct this**; it is your responsibility to submit.
- ***Plagiarism will not be tolerated!*** For information: GT Academic Honor Code and the Syllabus.
    - You will receive a 0 for the entire project; this includes both part 1 and part 2.
    - You will also be reported to OSI.
- Part 2 will have you crafting your own buffer overflow attack and will be released in a week.

Supplemental Readings:

- *Complete Virtual Memory Systems*
- *Memory API*
- *Address Spaces*
- If you want some GDB assistance there are plenty of helpful docs and tutorials.

Project Tasks (20 points):

I. Stack Overflow Toy Example - (20 points)

The year is 1988 and the Morris Worm is gaining media attention. It takes advantage of a technique called buffer overflow and, being young, you are delighted to try your hand at the technique. You find the login-verification source code for your school and discover a buffer overflow vulnerability. Testing the waters, you want to see if you can log into an account with higher privileges... for good, of course. Assume that there are no stack protections.

```
01.  #include <stdio.h>
02.  #include <string.h>
03.  #include "passwords.h" // Contains the global variable adminpass
04.
05.  int main (int argc, char *argv [])
06.  {
07.      int access = 0;
08.      char password[8];
09.
10.      printf("Please enter a password: ");
11.      scanf("%s", password);
12.
13.      if (strncmp (password, adminpass, 8) == 0)
14.          access = 1;
15.      if (access > 0)
16.          printf("Access Granted!\n");
17.      return 0;
18.  }
```

In the C language integers are 4 bytes and characters are 1 byte. We are considering a 32 bit system in which registers take up 4 bytes on the stack. Using this and the knowledge you acquired about stack layout (from the textbook) you construct the following rough diagram of the code's stack. This is the stack as the function is entered before any user input has been taken. The question marks indicate garbage data that has not yet been overwritten.

Register / Address	Variable	Value
EBP + 12	argv	?
EBP + 8	argc	?
EBP + 4	return address	?
EBP	frame pointer	?
EBP - 4	access	0
EBP - 8	password[4-7]	?
EBP -12 / ESP	password[0-3]	?

Complete the "Project 1 - Part 1: Stack Overflow Case Study" Toy Example Quiz on canvas with your responses to the questions regarding this task.

II. Understanding Countermeasures - (0 points)

Now that you've seen a toy example of a buffer overflow, it's important to explore some of the countermeasures against these sorts of attacks.

1. Address Space Layout Randomization (ASLR)
    - What is ASLR? How does it affect the stack?
    - Can ASLR be bypassed without turning it off? If so, how?
2. Stack Canary
    - What is a Stack Canary? How does it affect the stack?
    - Are Stack Canaries vulnerable and if so, how?
3. Do strongly typed languages suffer from the same buffer overflow vulnerability as in the toy example (which is weakly typed)? Explain why or why not.
4. With the toy example we saw a vulnerable function that allowed an overflow to take place; how could we make that function call less vulnerable or what other, more safe, functions could we replace it with?

III. Understanding Foundational Concepts - (0 points)

In order to complete the second part of the project, where you'll be performing your own buffer overflow, you need to become familiar with a few things.

5. What is the GNU compiler and GDB? How do you invoke both?
6. How would you use GDB to do the following:
    a. View the processor registers
    b. Set a breakpoint in code
    c. Find the address of an OS function (Such as a System level call like "exit")
    d. Inspect a memory location

    What other useful things can GDB be used for?
7. Give an explanation of the following gcc compiler flags. What are they and what do they do? How do they affect the program/binary and/or circumvent protections?
    a. `-g`
    b. `-fno-stack-protector`

    There are many more flags; discuss some you think might be useful!
8. What are some registers used by the stack? What is their purpose?

Join the appropriate TA-created edStem threads and discuss these questions with your fellow students. Even though these tasks are not worth any points, these topics will greatly assist you in completing Part 2 of the project. Discussion isn't limited to these topics either, so long as it doesn't pertain to any of the questions that we are grading you on. You'll miss out if you don't join in!