Application Software Security

Homework 01

Acknowledge any help and do not google for solution

Submission Instruction

Failure to follow the below instructions will result in a 20% penalty.

1. Create a folder named *"<First_name>_<Last_name>_HW_XX"* and put all your answers (e.g, source files) in this folder. For example: folder name *"David_Smith_HW_01"* for student name "David Smith" and assignment 01. No other object files or test files should be included.

2. Create a ZIP file of this folder with the same name (i.e., *"David_Smith_HW_01.zip"*) and submit it over Blackboard.

Question 01: Finish and turn in your C programs for Assignments 01 and 02 at the end of Lecture 01.

Question 02: Write a "Hello World" C program with some extra initialized and uninitialized variables (This program will further be used in other questions). When you compile it using gcc

(a) What is the effect of the option -Wall?

(b) What is the effect of the option -pedantic?

(c) What is the difference in the size of the code compiled with and without -g?

(d) What is the difference in the size of the code compiled with -O1, -O2, and -O3?

Question 03: Open the "Hello World" C program in the debugger *gdb*. Set a breakpoint at the first line. Run the program to that point.

(a) What is the current ESP?

(b) What is the current EBP?

(c) What is the current EIP?

(d) Use the *examine* or *x* command to show the assembly language code for the next few commands to be executed.

Question 04: With the "Hello World" C program

(a) Use the *strace* command to list all the system calls that your program makes.

(b) Use *objdump* to find the address of the following sections in your program: (a) .text (b) .bss (c) .data

Question 05: Rewrite the *"helloASM.asm"* program described in class. Compile and link it.

(a) Include the source, object, and executable code.

(b) Debug the program. What memory addresses contain the program's environment strings?

(c) Debug the program. What memory address contains the program's arguments?

(d) Debug the program and stop it immediately before the syscall for write.

- What is the current ESP?
- What is the current EBP?
- What is the current EIP?

(e) Debug the program. Where is the message string located?

(f) What is the return value for the write syscall? Where is it located?

Question 06: Write a C program that contains a stack-based buffer overflow. Explain in detail why the program has a stack-based buffer overflow flaw. Demonstrate the flaw by causing the program to crash with a segmentation fault. Include the state of the stack before the crash and determine exactly why the program crashed.

Question 07: Write a C program that contains a stack-based buffer overflow and make it SUID root. Run the program outside of the debugger as an unprivileged user and exploit the overflow to obtain a root shell. Include a description of exactly how the program was exploited, and a screen shot showing that a root shell was obtained. (Note: The lab will help)

Question 08: Give an example of a program that contains an integer overflow error and explain the problem. Correct the flaw by implementing appropriate range checking. Do not change the types (int, unsigned int, etc) of any of the variables.