System Security, Spring 2010

Exercise 4

Distribution: 16.04.2010
Hand in: 22.04.2010

Introduction

This exercise session will help you to gain understanding of calling stack frames and buffer overflow basics. It will also provide you with practical experience on how to use a debugger (gdb). The successful completion of this exercise will help you during the next exercise session, where you will be required to perform buffer overflow attacks.

- The tar-archive `vulnapp_ex4.tar` on the website contains a binary that you should use for this exercise. Furthermore, you will probably need a tutorial on GDB, such as the one available at http://sourceware.org/gdb/current/onlinedocs/gdb.html
- The binary can be run on Linux (tested on Ubuntu 8.10). If you want to do the exercise at home and you do not have a Linux system there, you will find a live CD image on http://releases.ubuntu.com/8.10/. It contains all necessary tools to complete this exercise. You may consider running it under a virtual machine (e.g., VirtualBox).

For this exercise, you can also use the Linux machines in CAB H56 and H57.

Important: You might need to set up the maximum size of "core-dumps" to 0 (in a BASH-shell: `ulimit -c 0`) in order to save space or remember to delete them after the completion of the exercise.

The VULNAPP Program

The VULNAPP program provides two ways to read user input. The first is by supplying the user input as a program argument in the command line. The second is provided by the program that asks for user input during execution.

In particular, the program is only vulnerable during reading user input as a program argument. The function `cpybuf` handles the copying of the user input in an insecure way.

Stack frames

1. Analyze the stack before and after calling `cpybuf` using gdb. To analyze the stack during the program flow you must set appropriate breakpoints and run the program. By using the commands `info stack` and `info frame` you can examine the stack and the stack frame respectively. What is the output after the commands and what is the meaning of it?

2. Use the information mentioned above and describe the stack frame of `cpybuf` straight after calling `strcpy`. More precisely, draw a diagram of the contents of the stack frame of `cpybuf` for string input: **ABCDEFGH**.

   What is the size of the vulnerable buffer in bytes?

Stack overflow

1. Experiment with different inputs and describe the outputs. Of course we are especially interested in error messages of stack overflows.
2. Describe what happened if the following errors occur after a function call:
   (a) Segmentation Fault
   (b) Illegal Instruction

Buffer overflows, Setuid-bit

Buffer overflows are mentioned often in combination with programs, in which the setuid-bit is set.

1. What happens if a program, which has this bit set, is executed?
2. Why do you think such programs are especially "dangerous"?
3. Would it be a good solution to simply not allow files with the setuid-bit set? Please argue.

Exploits

1. With an exploit we mean a piece of software which utilizes a weak point in a computer system. Suppose you learn of a weak point in a program which is not controlling the length of an input. Which steps are necessary to write an exploit for the program? (Describe informally the approach of an attacker).