EEL4930/EEL5934 Software Reverse Engineering for Security Course

Assignment 2

In this assignment you will practice some of the memory vulnerability exploits. Please support your answers with screen shots and drawing when applicable to get full credit.

1. (8 pts) Perform an attack on the passwordCheckEasy or passwordCheckEasyStdin program such that it terminates with a ***Login successful*** message even though the provided command line arguments do not include any of the hard coded passwords. Explain your answer by providing the command line argument and explaining the attack details.
2. (9 pts) Perform an exploit on the toy phone application (phone.c) such that the secret key entered by the user is replaced with another string, which later gets used when secure connection is established. Note that the phone application is a simple menu-based program that receives a sequence of commands from the command line. There are three types of commands: 1) k1, k2, and k3 for secret key related events, 2) f1, f2, and f3 for social media feeds, and 3) r1, r2, and r3 for robot commands. Among these commands, k1, f1, and r1 take an additional string argument to denote the secret key, the social media feed, and the command for the robot, respectively. Note that you can issue the commands on the command line in any order as long as you issue them as explained below.

   k1 SKArg: record the secret key SKArg

   k2 : establish a secure connection

   k3 : disconnect

   f1 FArg: record the feed FArg

   f2 : display the social media feed

   f3 : close social media app

   r1 CArg : send command CArg to the robot

   r2 : flush the robot commands

   r3 : close the smart robot app

   Your goal is to find a sequence of command line arguments such that secure communication will not be established with any of the keys entered using the k1 command and instead another string that you craft, e.g., the robot command, will be used. Explain your answer by providing the individual steps of the attack.
3. (8 pts) Inspect the binary compiled from RBPInDanger.c and explain what would be the side effect of overwriting the RBP value within the swap function due to a buffer overflow? Is it possible to cause a change in the program semantics without a crash? To get full credit, explain by referring to both the relevant parts of the binary and other information such as the state of the stack or the heap at runtime.