Homework Number: hw10
Name: Shu Hwai Teoh
ECN Login: teoh0
Due Date: Thursday 4/09/2020 at 4:29PM

1. specially crafted buffer overflow string:
   AAAAAAAAAAAAAAAAAAAAAAAAAAAA\xc8\x0d\x40\x00\x00\x00\x00\x00
2. explanation of why you chose the string:
   - address of variable str in clientComm(): 0x7fffffffe42b
   - return address of clientComm(): 0x7fffffffe448
   - entry to the object code for the secretFunction() function: 0x0000000000400dc8
   - difference between address of variable str and return address of clientComm(): 0x7fffffffe448 - 0x7fffffffe42b = 29
   - Therefore, the specially crafted buffer overflow string is 29*A concatenated with the entry to the object code for the secretFunction() function.
3. explanation of your fixes to the code: when the number of characters inputted by client is larger than MAX_DATA_SIZE, let the server send string “less” to client and prevent the server from using strcpy().
4. the modified parts of the server code by highlighting or underlining them: as the following page, the modified part is highlighted in pink.

```c
// Homework Number:  hw10
// Name: Shu Hwai Teoh
// ECN Login: teoh0
// Due Date: Thursday 4/09/2020 at 4:29PM

/*
/ This is a server socket program that echos recieved messages
/ from the client.c program. where 8000 is the port you want your server to monitor.
*/

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <arpa/inet.h>
#include <unistd.h>

#define MAX_PENDING 10     /* maximun # of pending for connection */
#define MAX_DATA_SIZE 5

int DataPrint(char *recvBuff, int numBytes);
char* clientComm(int clntSockfd,int * senderBuffSize_addr, int * optlen_addr);

int main(int argc, char *argv[])
{
    if (argc < 2) {
    fprintf(stderr,"ERROR, no port provided\n");
    exit(1);
    }
    int PORT = atoi(argv[1]);


    int senderBuffSize;
    int servSockfd, clntSockfd;
    struct sockaddr_in sevrAddr;
    struct sockaddr_in clntAddr;
    int clntLen;
    socklen_t optlen = sizeof senderBuffSize;

    /* make socket */
    if ((servSockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("sock failed");
        exit(1);
    }

    /* set IP address and port */
    sevrAddr.sin_family = AF_INET;
    sevrAddr.sin_port = htons(PORT);
    sevrAddr.sin_addr.s_addr = INADDR_ANY;
    bzero(&(sevrAddr.sin_zero), 8);

    if (bind(servSockfd, (struct sockaddr *)&sevrAddr,
                sizeof(struct sockaddr)) == -1) {
        perror("bind failed");
        exit(1);
    }

    if (listen(servSockfd, MAX_PENDING) == -1) {
        perror("listen failed");
        exit(1);
    }

    while(1) {
        clntLen = sizeof(struct sockaddr_in);
        if ((clntSockfd = accept(servSockfd, (struct sockaddr *) &clntAddr,
        &clntLen)) == -1) {
            perror("accept failed");
            exit(1);
        }
```

```
73
74          printf("Connected from %s\n", inet_ntoa(clntAddr.sin_addr));
75
76          if (send(clntSockfd, "Connected!!!\n", strlen("Connected!!!\n"), 0) == -1) {
77              perror("send failed");
78              close(clntSockfd);
79              exit(1);
80          }
81
82          /* repeat for one client service */
83          while(1) {
84              free(clientComm(clntSockfd, &senderBuffSize, &optlen));
85          }
86
87          close(clntSockfd);
88          exit(1);
89      }
90  }
91
92  char * clientComm(int clntSockfd,int * senderBuffSize_addr, int * optlen_addr){
93      char *recvBuff; /* recv data buffer */
94      int numBytes = 0;
95      char str[MAX_DATA_SIZE] = "less";
96      /* recv data from the client */
97      getsockopt(clntSockfd, SOL_SOCKET,SO_SNDBUF, senderBuffSize_addr, optlen_addr);
        /* check sender buffer size */
98      recvBuff = malloc((*senderBuffSize_addr) * sizeof (char));
99
100     if ((numBytes = recv(clntSockfd, recvBuff, *senderBuffSize_addr, 0)) == -1) {
101         perror("recv failed");
102         exit(1);
103     }
104
105     // if the number of received byte is larger than MAX_DATA_SIZE
106     // avoid using strcpy for variable str.
107     while(numBytes >= MAX_DATA_SIZE){
108         fprintf(stderr,"Too many characters for input will cause buffer overflow.\n");
109
110         //send "less" to let client input less characters
111         if (send(clntSockfd, str, strlen(str), 0) == -1) {
112             perror("send failed");
113             close(clntSockfd);
114             exit(1);
115         }
116
117         if ((numBytes = recv(clntSockfd, recvBuff, *senderBuffSize_addr, 0)) == -1) {
118             perror("recv failed");
119             exit(1);
120         }
121     }
122
123     recvBuff[numBytes] = '\0';
124     if(DataPrint(recvBuff, numBytes)){
125         fprintf(stderr,"ERROR, no way to print out\n");
126         exit(1);
127     }
128
129     strcpy(str, recvBuff);
130
131     /* send data to the client */
132     if (send(clntSockfd, str, strlen(str), 0) == -1) {
133         perror("send failed");
134         close(clntSockfd);
135         exit(1);
136     }
137
138
139     return recvBuff;
140 }
141
142 void secretFunction(){
143     printf("You weren't supposed to get here!\n");
144     exit(1);
```

```
145   }
146
147   int DataPrint(char *recvBuff, int numBytes) {
148       printf("RECEIVED: %s", recvBuff);
149       printf("RECEIVED BYTES: %d\n\n", numBytes);
150       return(0);
151   }
152
```