Homework Number: 10
Name: Yi Qiao
ECN Login: qiao22
Due Date: 04/04/2019

Buffer Overflow Attacks

My buffer overflow string:

AAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x3B\x4F\x55\x55\x55\x55\x00\x00

Explanation

The string consists of two parts. The first 29 "A"s and following the 8 bytes desired address. In the server's code, the size of the "str" buffer is 5 and anything beyond 5 characters will leads to a buffer overflow. So, we want the overflow overwrites the return address of the current call frame. From GDB, I get &str at 0x7fffffffd6fb and (unsigned*) $rbp at 0x7fffffffd710. Thus, the return address must live in 0x7fffffffd718. the difference between the return address and &str is 29, thus there are 29 "A"s and follows the desired address in a byte-reversed fashion.

Code

```
/**********************************
 *   Homework Number:   10         *
 *   Name:              Yi Qiao    *
 *   ECN Login:         qiao22     *
 *   Due Date:          4/4/2019   *
 **********************************/

/*
/ file : server.c
/----------------------------------------
/ This is a server socket program that echos recieved messages
/ from the client.c program.  Run the server on one of the ECN
/ machines and the client on your laptop.
*/

// For compiling this file:
//        Linux:              gcc server.c -o server
//        Solaris:            gcc server.c -o server -lsocket

// For running the server program:
//
//                server 9000
//
// where 9000 is the port you want your server to monitor.  Of course,
// this can be any high-numbered that is not currently being used by others.
```

```c
#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);
        }

        printf("Connected from %s\n", inet_ntoa(clntAddr.sin_addr));

        if (send(clntSockfd, "Connected!!!\n", strlen("Connected!!!\n"), 0) ==
        ↪  -1) {
            perror("send failed");
            close(clntSockfd);
            exit(1);
        }

        /* repeat for one client service */
        while(1) {
            free(clientComm(clntSockfd, &senderBuffSize, &optlen));
        }

        close(clntSockfd);
        exit(1);
    }
}

char * clientComm(int clntSockfd,int * senderBuffSize_addr, int * optlen_addr){
    char *recvBuff; /* recv data buffer */
    int numBytes = 0;
    //char str[MAX_DATA_SIZE];
    /* recv data from the client */
    getsockopt(clntSockfd, SOL_SOCKET,SO_SNDBUF, senderBuffSize_addr,
    ↪  optlen_addr); /* check sender buffer size */
    recvBuff = malloc((*senderBuffSize_addr) * sizeof (char));

    if ((numBytes = recv(clntSockfd, recvBuff, *senderBuffSize_addr, 0)) == -1) {
        perror("recv failed");
```

```
        exit(1);
    }

    recvBuff[numBytes] = '\0';
    if(DataPrint(recvBuff, numBytes)){
        fprintf(stderr,"ERROR, no way to print out\n");
        exit(1);
    }

    //strcpy(str, recvBuff);

    /* send data to the client */
    if (send(clntSockfd, recvBuff, strlen(recvBuff), 0) == -1) {
        perror("send failed");
        close(clntSockfd);
        exit(1);
    }


    return recvBuff;
}

void secretFunction(){
    printf("You weren't supposed to get here!\n");
    exit(1);
}

int DataPrint(char *recvBuff, int numBytes) {
    printf("RECEIVED: %s", recvBuff);
    printf("RECEIVED BYTES: %d\n\n", numBytes);
    return(0);
}
```