Terrence Randall
Randall7
4/9/20

Specially crafted string for overflow attack:

The string is:

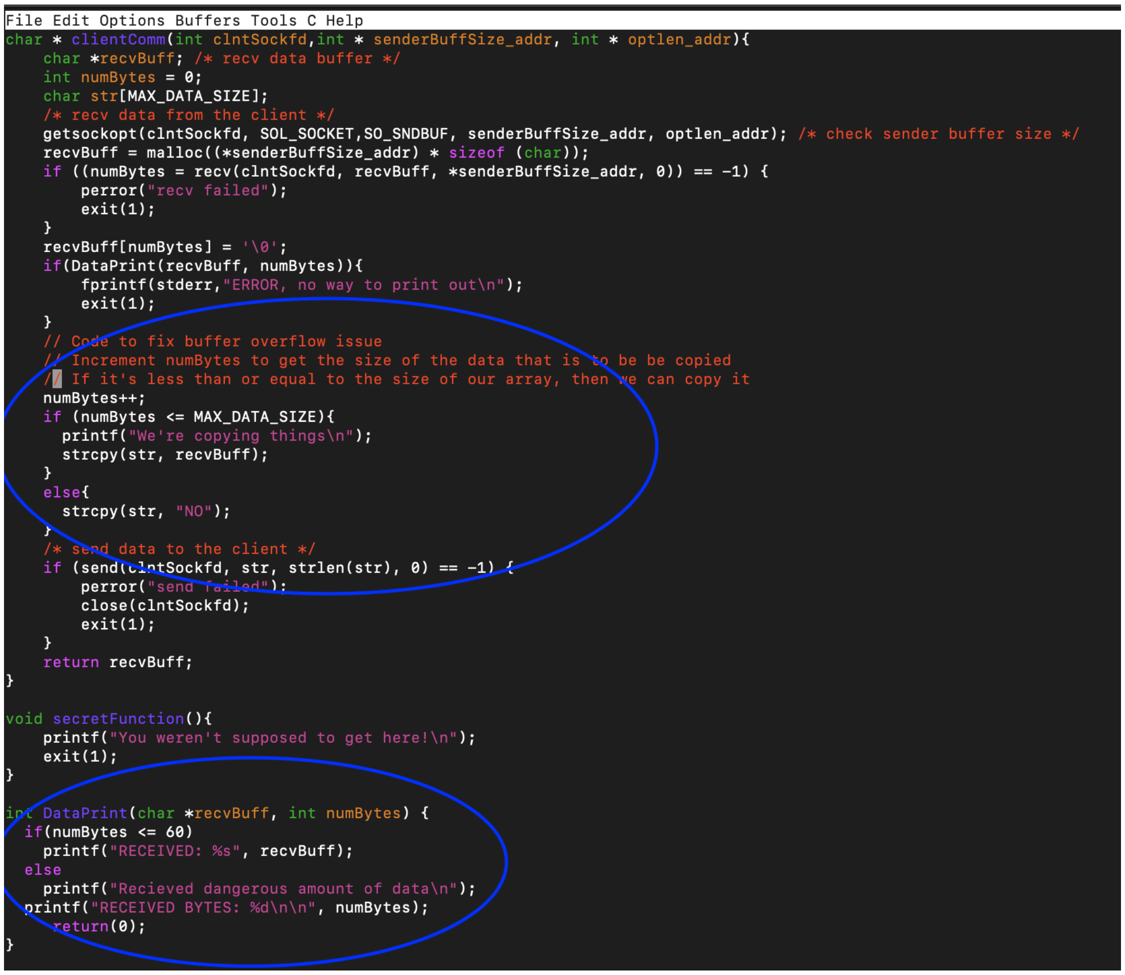
```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\xc9\x0d\x40\x
00
```

Rationale behind decision:

A certain amount of characters are required as an input into the stack before we can get to the location where we're overwriting the return address for the current frame. For our example, it took 56 As until we got to the location where we could overwrite the return address with the next 4 input bytes.

Terrence Randall
Randall7
4/9/20

Modified Server Code:



```c
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);
    }
    // Code to fix buffer overflow issue
    // Increment numBytes to get the size of the data that is to be be copied
    // If it's less than or equal to the size of our array, then we can copy it
    numBytes++;
    if (numBytes <= MAX_DATA_SIZE){
      printf("We're copying things\n");
      strcpy(str, recvBuff);
    }
    else{
      strcpy(str, "NO");
    }
    /* 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) {
  if(numBytes <= 60)
    printf("RECEIVED: %s", recvBuff);
  else
    printf("Recieved dangerous amount of data\n");
  printf("RECEIVED BYTES: %d\n\n", numBytes);
     return(0);
}
```

Notes on Changes made:

The initial vulnerability was present because there was no check on the size of the data being taken in as an input. And if the size was bigger than the area allocated on the stack, when the input data is copied over to the stack variable it will overwrite data beyond its bound. This means that with the right input string an attacker can overwrite the return address of the function to be the entry point into the "secretFunction".

To fix this vulnerability, I first added an increment to "numBytes" so that it would then be the size of the array that was attempting to be copied. After the increment, I checked that the value of numBytes was less than or equal to the size of the array. If it is then the program may execute the string copy, otherwise it'll copy a string that says "no" to tell the user that they sent too much (and so that the server program doesn't SegFault). Further, at the bottom there are alterations to DataPrint to tell the server what happened.