First ss: Disas secret function and compilation

Text

Description automatically generated

Second ss: Disas clientComm

The second step of the buffer overflow attack was disassembling the clientComm function and printing a stack trace. This was done primariy to find the leaveq address so we could set a breakpoint later. Two breakpoints were set in total, one at the start of the clientComm function and one at the leaveq address.

The server was started at the port 9000 and the client server was directed to the ip address at the same time. Thus, the server established a connection with the client.

Graphical user interface, text

Description automatically generated

Third ss:

Ran the server with the breakpoints, and successfully sent 3 As as a test. Then, I printed the return address for the stackframe of clientComm. Then, starting at the location of the stack pointer, I displayed a segment of 80 bytes and looked for the vlue 0x41 (A in hexadecimal). I counted the number of bytes between the first appearance of 0x41 and the return address of the stackframe.

Text

Description automatically generated

Fourth ss:

Then, on the client side, I added the first address of SecretFunction to the end of a stream of As. The number of As is equal to the number of bytes calculated in the last ss. In this case, it was 29. The address of secretFunction was reversed before appending it.



This led to the secretFunction being called and the buffer overflow attack being successful.

Text

Description automatically generated

For finding the vulnerability, I carefully combed through the clientComm function and I noticed that the string copy function receiving from the buffer was just copying everything without a limit on the data ize. Therefore, I changed it to strncpy instead and added the MAX\_DATA\_SIZE parameter.

