A screen shot of a computer

Description automatically generated

Process:

* Put a breakpoint at start of ClientComm function
* Disassembled clientComm

A screen shot of a computer program

Description automatically generated

* Found the address of leaveq in memory, which is what clientComm is going to return
* Put a breakpoint there
* Run, and stop at breakpoint 1, and next from there until we get to strcpy
* Here, print mem address of str, because that is where the value we get to input goes
  + Remember that value

A computer screen shot of a code

Description automatically generated

* Get to breakpoint 2, which is the end of the function, and print the return address of the stack frame. This is where we want to overflow a value into.
* Take the difference of these two mem addresses. That is how far we need to offset in our buffer overflow

A screen shot of a computer program

Description automatically generated

* That difference is decimal 40, so we send 40 characters, then the reversed address of secretFunction, which we got from disassembling it earlier. This puts the address of secretFunction in the return address of clientComm, so when clientComm ends it doesn’t return properly and instead execution goes to secretFunction.
  + Overflow string is AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\x40\x00

In order to fix this vulnerability, I added an if block before the strcpy to check that the buffer did not exceed the max data size used to declare str. This causes an error if an overflow attack is attempted.

A computer screen shot of text

Description automatically generated