Echo Server Exploit
Cpt S 422 Homework Assignment
by Evan Olds

This is a coding assignment. You must submit zipped C++ code that compiles in Linux with the build script included in the assignment zip.

You may work in groups of up to 3 people on this assignment. Groups of more than 3 are not allowed. The assignment requirements are the same regardless of group size (so it's in your best interest to work in groups of 3!). Submit 1 copy of the homework per group. Make sure names of all the group members are on the assignment. Submit a .zip online with code files containing implementations for the requirements listed below.

Open the code project included in the assignment .zip file. Your job is to engineer the exploit that we discussed in class, where the client tricks the server into sending it the contents of the pwd.txt.

Rules:

- Modify only the client app code
- Do not modify the server code
- Assume the server is running on a remote machine, although for testing purposes running the client and server on the same computer is fine
- Assume the server has the pwd.txt file in its working directory

This assignment is mostly about figuring out where the weak points are in the server code. Your client app code will exploit these vulnerabilities in order to get the password file data. Spend time working with your group members to map out all the details and figure out how to engineer the exploit. Then write the code and submit it for the assignment.

Help:

All the assignment requirements are above and this section just contains a few helpful bits of information.

1. The socket sending functionality is such that it generally copies into the network buffer and returns immediately. It will block if the network buffer is full, but in general the send function is going to return quickly.
2. The socket receiving functionality is such that it blocks until incoming data arrives in the network buffer. This means that when the server makes a receive call it will sit there doing nothing if no data has been sent to it. This is a crucial part of the exploit.
3. The server has a pool of client structures and accepts new connections as long as it can find an unused structure in the pool. The **`InUse`** member of the structure is 1 if the client structure represents an active connection and 0 if it is available to use for a new connection. This is another crucial part of the exploit.
4. The exploit solution I showed in class involved creating multiple connections to the server at a time and had a bit of a back-and-forth until the proper server state was constructed and the desired data was sent. You should be engineering a similar exploit. Although, if you find an alternative way to make the exploit work, that will be accepted too. If your alternative way is easier (involves fewer steps) I may consider giving extra credit for the assignment.
5. This was already mentioned earlier, but it's worth reiterating that this assignment doesn't involve much code. It's more about studying the scenario and figuring out how everything works, then engineering the exploit. Spend time with group members to figure out how the existing code works, discover where the vulnerabilities are, and determine the implications of these vulnerabilities. Very little code is needed to do the exploit once you know how it's going to work.
6. The TCPComm class that's included provides some socket functionality for you. It has a callback function that you can set to a function that you want called when data is received. It has a send function that you can call to send data over the socket connection.