Alex M Brown
CS 472
Assignment 3 Questions

1. Consider the security of the data of FTP and the commands. Good or bad? Is BitTorrent better or worse? Why? What about SFTP? Is it the way that it is implemented or are there considerations because of how the protocol works?

FTP has pretty lackluster security. All of the commands are sent in plain-text, so anyone with a packet sniffer could read everything you send/receive from the server. This would allow someone to see the username and password that you used to connect to the server, and from there, you're sunk. They now have the same access to the server as you do.

BitTorrent's security isn't flawless either. The main issue with a Peer-to-Peer system like BitTorrent is the fact that you have to trust every peer thats connected. If even a single peer is malicious, the whole thing falls apart. For example, if I connected to a torrent and started sending malicious software under the guise of a file that everyone else was building, they would get (at best) a corrupted file or (at worst) my malicious script.

SFTP is much more secure than FTP because SFTP uses SSH. SFTP ensures that clients are only connecting to trusted servers and servers are only being accessed by trusted clients. Additionally, the meessages are encrypted, so packet sniffers can no longer read usernames and passwords.

With FTP, the risks come from how the protocol is implemented (plain-text messages) while with BitTorrent, the risks come from how the protocol works (forced trust with all peers).

2. Do you think that someone could hack into your FTP server? Why or why not? Through what methods?

Yes, I believe if someone really wanted to, they could hack into my FTP server. One way this could happen is if they were using Wireshark or some other packet sniffing software while I was using the server. They would be able to see my command messages in plain-text, including USER and PASS commands. Once logged in they could read any files in the FTP environment while uploading malicious files.

I don't think anyone would be able to mess with my server by sending it bad requests once connected. I implemented a parser that only runs requests if they are recognized and syntactically correct. If they were to send anything other than a legal command, they would just get an error response.

Additionally, I took extra precaution in making sure no one access anything outside of the FTP environment on the machine that the server is running on. You cannot change the directory out of the environment, get listing information of a file or directory outside the environment, store files outside the environment, nor retrieve files outside the environment.

These features make it a little sturdy, but I definitely wouldn't say its impenetrable.

3. Critique how FTP solves the problem of file transfer – what is good? What is bad? What is weird?

A good aspect of FTP is its simplicity. Every command follows a simple format of a (mostly) four letter command, a number of parameters separated by spaces, and then a CRLF to signify the end of the message. The response codes are also easy to understand, because FTP implements the same style of response codes as SMTP, where each number in each spot of the code has a specific meaning. For example, a code with a 2 in the hundreds spot represents a success, while a 5 represents a complete failure.

A bad aspect of FTP is its security. Everything is sent over ASCII, and none of it is encrypted. This includes passwords. Anyone running a packet sniffer during an FTP authentication would be able to read the PASS command in plain-text.

A weird (but arguably necessary) aspect of FTP is its dual connections. FTP uses one connection for receiving commands and sending responses and another connection for reading/sending data for files and directory listings. This is confusing when considering the transfer of information between the controller and the server, because why would you need to connect to something you are already connected to. However, this becomes very useful when a client is trying to send a file from one server to another. The server uses the data connection to connect to the second server.

4. How hard would it be to make FTP use on channel instead of two?

I would argue that it would be very difficult to change FTP so it only uses one channel. While it wouldn’t be too difficult for exchanging files between the server and the client, having only one channel would cause all sorts of problems when trying to exchange data between two different servers. In order to keep the ability to exchange data across servers from a client, the FTP servers would have to close their connections to the client every time they were exchanging data with each other. This could cause problems if the connections were to fail at some point, because the servers wouldn’t be able to reach the client anymore. The client would have to rely on a timeout to know if something went wrong, but that timeout would have to be very long for extremely large files. The client would be completely blind to the file transfer until the servers reconnected to the client.