BISON-Transfer Explained

Brandon Lu @ UCR UAS

14 March 2015

1 Introduction

This document describes the BISON-Transfer server and client suite's purpose, implementation, and methods of communication. If it is not clear by this point, the server (or daemon) is called `BISON-Transferd` and the client (client daemon, to be precise) is called `BISON-Transfer`.

2 Specifications

The BISON-Transfer suite is (hopefully) designed to operate within the following constraints:

- Timely image transfer
- Robust communications (disconnect handling)
- Efficient transfer protocol

Contents

3 Communication

At initialization, the transfer server fork is in an unknown state. It must recieve a transfer mode from the client or it will not know what to do and will terminate. After it has recieved a transfer mode, it can then proceed to either send the file, send the filetable, or recalculate and then send the md5 sum of a file.

4 Transfer Modes

There are four transfer modes that have been (or will be) implemented. These are presented in tabular form in 1.

`REQ:␣`<filename>`␤␤`	File Request
`FTREQ␤␤`	Filetable Request
`RECALC:␣`<filename>`␤␤`	Request MD5 Recalculation

Figure 1: Transfer Modes for the BISON-Transfer suite

Please note that the (␤) character is meant to represent ASCII newline (decimal `10`, hexadecimal `0x0A`). The visible space (␣) is meant to notate an ASCII space (decimal `32`, hexadecimal `0x20`). It is simply convenient to have these symbols explicitly represented in order to avoid confusion.

It should also be noted that the <filename> should be entirely replaced with the corresponding filename that is going to be transferred.

4.1 File Request

The file request mode is signified by the client querying the server for the file by providing the command in figure 2.

REQ:␣<filename>↵↵

Figure 2: Request File Mode

The server, after recieving this line, will provide the corresponding file in unencrypted binary over the port.

4.2 Filetable Request

The client requests the filetable using the following line:

FTREQ↵↵

Figure 3: Filetable Request Mode

The filetable format is done by line, and files are not expected to have newlines in their names. The format is provided in figure 4.

<MD5 SUM (32 hex-representing ASCII characters)>␣␣<filename>↵

Figure 4: MD5 Sum and Filename Specification

The last line of the filetable will have a newline as well. This is followed by the server terminating the connection.

4.3 Request Filetable Recalculation

Ideally, this specific transfer mode is never used because it signifies an error either in transmitting the filetable or calculating the MD5 Sum on the transfer server. The client requests a filetable recalculation using the following line:

RECALC:␣<filename>↵↵

Figure 5: Request Recalculation

This makes the server recalculate the MD5 sum of a specific file and not the entire directory. Usually the MD5 sums of the entire directory are calculated on server startup. A new MD5 sum is only calculated when a new file appears.

Old MD5 sums are not recalculated when the file is changed, so it is not wise to change files within the transfer directory while the server is running. Ideally, a solution to this would be to recalculate the MD5 sum when the "mtime" of a file is changed.

5 Client State Machines

To be written.

6 Security

In a closed-system, security is not generally considered for simplicity sake. This is not a system designed for large-scale deployment by any means. It is designed to work for its designated purpose, and only that purpose. Any other use of it is probably unsafe, and the author does not have grandiose visions of the software's use anywhere other than transferring trivial (by volume) files such as the images acquired from an unmanned aerial vehicle built by students from UCR.

But considering security for the heck of it is an interesting topic in general. This is especially true when it comes to code-masochists like the author.

6.1 Filenames

Filenames should not contain newlines for sanity sake.

A file with a newline will cause the program to either reject the file or exit on error condition.

6.2 Random Characters

Random characters from the server under file transfer mode can potentially cause the BISON-Transfer daemon to overload the filesystem on the client side. This issue is not addressed because it will not happen unless an unsafe server masquerades as `BISON-Transferd`.

Random characters entering the command parsing state machine from an attacker client will cause the corresponding thread on the server to abort.

It is noted that the client does not reject random characters from the server before it issues a command. This may or may not be resolved at a later date.

7 Performance

Transfer overhead is fairly low, and mostly depends on underlying hardware and TCP/IP stack.

The protocol simply asks the full file name and expects to get the file back.