GRep-AS-a-Service
GRASS is buggier on the other side

Short-version, 04/06/2020

1 Introduction

"GRASS" – GRep AS a Service, defines a protocol for an online search and file management service. Here, we provide you with an implementation of GRASS. For the sake of brevity, we will refer to both the protocol *and* the implementation by the name GRASS hereon. In the spirit of `ssh` and `ftp`, GRASS implements a clear-text service that allows remote command execution, file transfer, and search. For compartmentalization, our protocol separates command and data channels. To simplify the implementation, we assume that the network is trusted. In practice, we would transfer both control information and data through an encrypted channel.

The protocol implements several commands: `login`, `pass`, `ls`, `cd`, `mkdir`, `rm`, `get`, `put`, `search`, `date`, `whoami`, `w`, `ping`, `logout`, and `exit`. For the purpose of an example bug, the protocol adds another instruction: `fault`. The command channel is implemented as a simple bidirectional TCP connection. The data channel is implemented as a second TCP channel to a dynamic port.

1.1 Commands

Some commands may be available even without login, others are only available to logged in users, e.g., listing, accessing, or changing files. After authentication, the user can then execute any of the commands. All commands must be followed by a Unix newline ($\backslash n$). The server signals any error by starting the response with `Error`, followed by a human-readable error message and a newline.

login The `login` command starts authentication. The format is `login $USERNAME`, followed by a newline. The username must be one of the allowed usernames in the configuration file.

pass The `pass` command must *directly* follow the `login` command. The format is `pass $PASSWORD`, followed by a newline. The password must match the password for the earlier specified user. If the password matches, the user is successfully authenticated.

ping The `ping` may always be executed even if the user is not authenticated. The `ping` command takes one parameter, the host of the machine that is about to be pinged (`ping $HOST`). The server will respond with the output of the Unix command `ping $HOST -c 1`.

ls The `ls` command may only be executed after a successful authentication. The `ls` command (`ls`) takes no parameters and lists the available files in the current working directory in the format as reported by `ls -l`.

cd The `cd` command may only be executed after a successful authentication. The `cd` command takes exactly one parameter (`cd $DIRECTORY`) and changes the current working directory to the specified one.

mkdir The `mkdir` command may only be executed after a successful authentication. The `mkdir` command takes exactly one parameter (`mkdir $DIRECTORY`) and creates a new directory with the specified name in the current working directory. If a file or directory with the specified name already exists this commands returns an error.

rm The `rm` command may only be executed after a successful authentication. The `rm` command takes exactly one parameter (`rm $NAME`) and deletes the file or directory with the specified name in the current working directory.

get The `get` command may only be executed after a successful authentication. The `get` command takes exactly one parameter (`get $FILENAME`) and retrieves a file from the *current* working directory. The server responds to this command with a TCP port and the file size (in ASCII decimal) in the following format: `get port: $PORT size: $FILESIZE` (single space between words, followed by a newline) where the client can connect to retrieve the file. In this instance, the server will spawn a thread to send the file to the clients receiving thread as seen in Figure 1.

The server may only leave one port open per client. Note that client and server must handle failure conditions, e.g., if the client issues another `get` or `put` request, the server will only handle the new request and ignore (or drop) any stale ones.

put The `put` command may only be executed after a successful authentication. The `put` command takes exactly two parameters (`put $FILENAME $SIZE`) and sends the specified file from the current local working directory (i.e., where the client was started) to the server.

The server responds to this command with a TCP port (in ASCII decimal) in the following format: `put port: $PORT` (also single spaces, followed by a newline). In this instance, the server will spawn a thread to receive the file from the clients sending thread as seen in Figure 2.

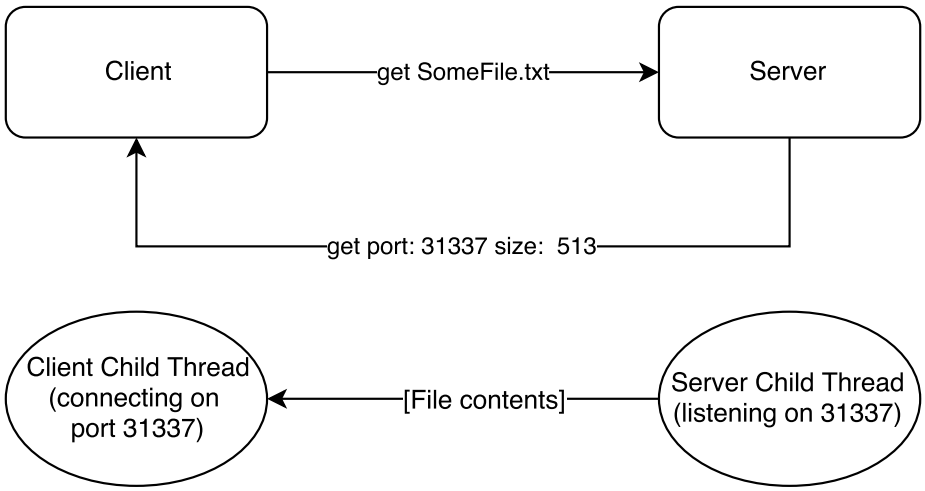

Figure 1: Get Command

grep The grep command may only be executed after a successful authentication. The grep command takes exactly one parameter (`grep $PATTERN`) and searches every file in the current directory and its subdirectory for the requested pattern. The pattern follows the Regular Expressions rules[1].

The server responds to this command with a line separated list of addresses for matching files in the following format: `$FILEADDRESS $ENDLINE`.

date The `date` command may only be executed after a successful authentication. The `date` command takes no parameters and returns the output from the Unix `date` command.

whoami The `whoami` command may only be executed after a successful authentication. The `whoami` command takes no parameters and returns the name of the currently logged in user.

w The `w` command may only be executed after a successful authentication. The `w` command takes no parameters and returns a list of each logged in user on a single line space separated.

logout The `logout` command may only be executed after a successful authentication. The `logout` command takes no parameters and logs the user out of her session.

[1] `https://www.gnu.org/software/grep/manual/html_node/Fundamental-Structure.html`

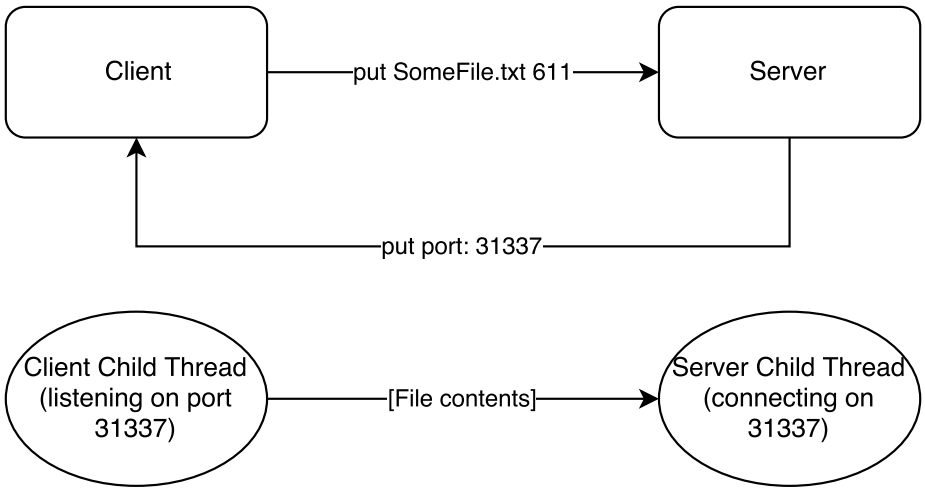

Figure 2: put Command

exit The `exit` command can always be executed and signals the end of the command session.

GRASS allows for `put` and `get` to be executed in parallel, with multiple parallel uploads and downloads supported at the same time.

1.2 Configuration file

The configuration file (`grass.conf`) specifies the runtime configuration for the server. The following directives are supported: base, port, and user. The `grass.conf` file needs to be placed in the same directory as the client. An example configuration is as follows:

```
# Grass configuration file
# # marks a comment.

# Current directory is the base directory
base .

# Format: port number
port 1337

# Format: user name pass
user AcidBurn CrashOverride
```

The server takes exactly no command line arguments

```
./server
```

The client takes 2 command line arguments: the server ip and port:

```
./client server-ip server-port
```

1.3 Error

The software signals an error by sending a response with `Error`, followed by a human-readable error message and a newline. Both the sender and receiver of an error print the error.

Access The server starts the execution with the base directory. The server program restricts the user to the base directory. If a client attempts to get access to any file or directory outside of base directory, the server sends an error: "Error: access denied!"

File name The server must support path names of up to 128 character from the root of the base directory. If a file path goes beyond this limit, the server sends an error: "Error: the path is too long."

Transfer When uploading or downloading a file, if the transferred stream's size doesn't match with the specified size in commands `get` and `put`, the program sends an error: "Error: file transfer failed."