GFTP – General File Transfer Protocol

Status of this memo

This RFC suggests a proposed protocol for private persons in contact with the author of this RFC. This document is therefore not for public consumption. Distribution of this memo is forbidden.

Overview

This RFC describes an IP/TCP general file transfer protocol (GFTP) which allows a client to query the host for a list of files and them to exchange files between them. The general file transfer protocol operation contains 3 different operations: a list query for the server, downloading a file to server and uploading a file to server. This RFC describes all of them.

Protocol Operations

Overview

All the operations are performed in 3 stages. The first stage is the basic TCP syn/ack sequence, which is followed by a TLS handshake, which is followed by the actual content of the operation. As protocol includes forming a TLS tunnel, it is a stateful protocol.

List Operation

After securing shared connection through TLS tunnel, a client will send a list request to the server, coupled with a PGP public key which the server will accept based on the server's own database with an equivalence check. The list command must be a utf-8 formatted string spelling "list" sent as bytes. No additional information is needed or accepted by the server. If the server matches the received public key in its database, the server will send a list of available files to the client. The list will include the name of the file including its type, and the size of the file. The minute details of the list are left to the implementer of the server, but it should at least contain the two aforementioned things. All the files will be stored on a certain folder on the server, and that's the only location the server will provide a list of or indeed, provide any action on.

Download Operation

After a TLS handshake, the client must send a public PGP file to the server. If the server accepts the key (provided that this public PGP key is in the server's repository), the client sends a file name which it wants to download to the server. The server only accepts a filename, it will not accept a filename with any path notation, eq. "/" or "\" -signs or "..". If the server detects the usage of such symbols, it will reject the request and sends an error message to the client. The server also sends an error message if the PGP key send by the client didn't match any PGP key on the server's repository.

In case of the server accepting the request, it will upload the file to the client. The uploading process begins with the server providing the size of file to the client. After that, it will begin to send the file in a 1024 byte - sequence. The server must signal that the transfer is over by sending a packet with the value of zero in string in it. The client stops writing to a file the instant it receives this message. After the file is transferred, the client will close the file, compare the size of the actual file to the size of the file proclaimed by the server, and if the sizes don't match, it will delete the file. It has to be noted, that with TCP, the file sizes should always be the same as the mechanisms of TCP ensure proper delivery, so the matching of sizes can be omitted by the client, and the server won't wait for a response of successful delivery.

The sending will fail if a file of that name could not be created (e.g. there already exists a file of that name in the directory or the filename is wrongly formatted or contains illegal characters).

Upload Operation

This operation is the same as the previous one, except now the client must send its private PGP key to the server. If the server has it in its repository, the server will permit uploading a file to its directory. The server will only permit a filename, it will reject any symbol that refers to the file path, for example "/" and "\" or "..". It will wait first for the size of the file, and then start receiving the file. The sending of the file follows the same process as described in the download operation. The sending will fail if a file of that name could not be created (e.g. there already exists a file of that name in the directory or the filename is wrongly formatted or contains illegal characters).

New PGP keys cannot be sent to the server with this operation, as the folder in which the keys are stored is not the same as the folder to which the files can be sent. Adding new PGP keys to the server is not included in this document and it is not a part of this protocol.

A file named "list" cannot exist in the server's repository. If such a file exists or a file by that name is attempted to be uploaded into the server, the server will send an error to the client (and delete the file by that name if it exists on the server's directory). This rule must be enforced so the server can differentiate between "list" command and an actual download command with a filename as a parameter.

Packet Format

The GFTP packet is enclosed in standard TCP protocol, any numeric fields are transferred in the standard order of the network, big endian.

In the IP header of any GFTP message, the client and server will use public IP addresses in source and destination fields. The method by which the server and the client obtain these is not part of this protocol, however the destination IP address will always be required in the client and server side of the protocol. It can be a private IP address, provided the client and the server reside in the same subnet.

In TCP header, all the normal fields of the TCP header are used as per the TCP protocol.

The message itself, residing in a TCP payload, is encoded in UTF-8 formatted string or it is handled as raw bytes, depending on the operation. In case of download and upload operation the actual file transfer itself is handled 'as is', that is bytes. However, everything else is a utf-8 formatted string converted to bytes while in transfer. That means that, for example, the filename and the file length will be send as utf-8 formatted string sent as bytes.

Client Use of the Protocol

The client's operating system must contain support for TCP/IP and TLS. In the case of downloading a file, the client must have enough available space in its memory. A PGP key stored in a file is required for successful operations. That key should match the key in the server’s cache. How the distribution of PGP keys for server and client is implemented is not included in this document.

Server Use of the Protocol

As with the client, the server’s operating system must contain support for TCP/IP and TLS. In the case of upload operation by a client, the client must have enough available space in its memory. It is required that the server will have a cache of PGP keys to verify that the client is permitted to perform the required operation. The creation of PGP keys is not included in this protocol. Finally, the server is required to listen for active connections and have a port open for it so it can accept the operations for the GFTP protocol. The port used can be freely chosen by the users of the protocol.