Network Working Group
Request for Comments: 501
NIC: 15718

K. Pogran
MIT-Multics
11 May 1973

Un-Muddling "Free File Transfer"

As the ARPA Network begin to mature, we find ourselves addressing issues and concepts deliberately put off and left untouched at earlier stages of Network development. Among the issues now coming to the fore are access control, user authentication, and accounting. These issues arise immediately out of efforts to develop uniform methods for providing limited "free" access to the File Transfer Servers of the host systems, to meet user needs for mail transmission and similar services.

Several proposals have been made, described by such phrases as "login-less mail", "'free' accounts", "free file transfer", etc. These proposals inevitably have imbedded in them some particular notion of how such things as access control and user authentication are accomplished and these proposals, which knowingly or unknowingly make presumptions about the implementation of such mechanisms, inevitably meet with strong criticism from implementors whose systems' mechanisms are quite different.

In RFC 467, Bob Bressler proposes ways of helping out users who wish to transfer files to or from "systems which have some flavor of security, but on which the user has no access privileges". Unfortunately, beginning with the first paragraph of the RFC, the notions of access controls on files (examples of protection mechanisms), and control of access to the system (user authentication) are thoroughly muddled. In addition, he makes sweeping assumptions about the nature and use of accounting mechanisms and accounts at server sites. RFC 487 also has buried deep within it assumptions about the nature of the access control and user authentication aspects of File Transfer Server implementations.

What's needed at this juncture, of course, is a lucid discussion of the general concepts involved in protection mechanisms, and file system access controls in particular. Well, you won't find that in the remainder of this RFC. What you will find is perhaps enough of a discussion to un-muddle that which RFC 487 has muddled; the rest will have to come down the pike at a later time.

In many systems, mechanisms which control access to the system, mechanism which control access to files, and accounting mechanisms all mesh at the moment at which a prospective user of the system is authenticated: the system has checked his user-name, password,

account, or whatever, and decided that he is, indeed, a valid user of the system. This user, who would like to have some information processing performed on his behalf, is termed a principal in "privacy and protection" parlance. Some number of processes are initially set up for this principal, and some (presumably unforgeable) principal identifier is associated with the process(es), so that their requests for access to files and other system resources may be properly validated. In addition, the identify of the account to be charged for the resources consumed by these processes is associated with the processes at this time [1], although on some systems, a process may change its account identifier at any time.

The first question is: Whose principal identifier does a File Transfer Server process use? There are at least two possibilities: 1) the File Transfer Server can run as a "system daemon" process, with (usually) a highly privileged principal identifier. When acting on behalf of a user, it must, itself, interpretively evaluate that user's access to a desired file. Also, it must be able to charge that user's account for the resources it uses. 2) A File Transfer Server process can be given the user's own principal identifier. With this implementation, validation of the user's access to files is performed automatically by the usual file system mechanisms.

Paragraph four of RFC 487 clearly presumes implementation 1): "If a user connects to an FTP server and makes a file request without supplying a user name-password, the server should then examine the file access parameters ..." Systems truly concerned about protection may prefer implementation 2), and for good reason -- it follows the "principle of least privilege", which states that a process should execute with as little access privilege as it requires to perform its tasks properly. Running a File Transfer Server process with a user's principal identifier rather than with that of a system daemon leaves the system far less susceptible to damage caused by incorrect actions of the File Transfer Server. [2]

The next question is: Whom do you charge for file transfers? Bressler tries to set down some guidelines for determining who to charge for "non-logged-in" (read: "free") file transfer usage: "Clearly, storing a file in a user's directory can be charged to that user." How is the word "storing" used here? Surely, "that user" can be billed for the disk or other storage medium charges incurred by that file which is now taking up space, but is it legitimate to charge "that user" for the I/O and/or CPU resources used by someone else to transfer a file over the Network, and place it into that user's directory? For example, should the recipient of Network mail be charged for the resources consumed by someone else in sending it? (Would you care to pay the postage for all the junk mail that arrives in your home (U.S. Mail) mailbox?).

Over the telephone, Bob explained to me that he desired a mechanism which would, for example, enable me, at his request, to transfer a file from my system to his directory on his system, without requiring that I know his password. All well and good. In this situation, it would make sense to charge Bressler's account for this file transfer. But how does an un-authenticated user tell a server "Charge this to Bressler's account because he says it's OK"? Pitfalls abound. The file Transfer Server process needs to be able to charge an arbitrary user's account; this again presupposes implementation 1) of the File Transfer Server described above (or else any user process in the system would have the capability of charging any user's account; this seems undesirable). A more reasonable approach would be to charge that instance of the File Transfer Server process to a general "Network services" account. Mechanisms for accomplishing this are presented in RFC 491. [3]

RFC 487 matter-of-factly suggests that retrieval of files in "system" directories should be charged to "overhead". Here too, some broad assumptions are made about the nature of accounting mechanisms and accounts at server sites. In addition, an undesirable loss of generality is imposed upon the File Transfer Server: It is now required to have the capability of distinguishing the pathnames of "system" files from those of "user" files. In a number of systems, there is no syntactic distinction between the two, and the same general mechanisms can be used to manipulate both kinds of files (if a distinction between them can be made at all). The addition of code to the File Transfer Server which examines the pathname given for each request, to determine which sort it is, seems to be antithetical to the goals of uniformity and generality that many of today's systems have achieved.

The statement that a Network user's file transfer activity can be charged to a system-wide "overhead" account contains two assumptions: Such an account cannot be presumed to exist on all systems; furthermore, if it does exist, in some cases it just isn't the right account to charge. Certainly, a good case can be made that the cost of fostering inter-user communication within the ARPA Network community (which is what "free" file transfer amounts to) should be borne by ARPA, meaning that such activity should be charged to ARPA-funded accounts. If a host system's operation is entirely funded by ARPA (or if its management doesn't care who pays for this activity), then it makes sense to charge "free" file transfer activity to a "system overhead" account. On the other hand, that isn't the correct course of action for a host system whose operation is not funded by ARPA, for charging "free" file transfers to "system overhead" would result in passing the cost along to local customers who may have no interest at all in the ARPA Network.

Lastly, Bressler suggests that for file retrieval, CPU charges "may be sufficiently small to not cause a major problem". To believe this is naivete. It may appear to be true for a system which doesn't charge the user for time spent executing supervisor code, or I/O routines, where Network software overhead doesn't show up in the user's bill. In this case, Network software overhead must contribute to "general system overhead", the cost of which must be borne by all users. I don't think many people in the Network community would consider the actual (as opposed to charged) CPU time spent transferring a file to be negligible. Certainly, if a system is a very popular or busy one from a Network standpoint, the cumulative CPU time spent on "free" file transfers, viewed at the end of an accounting period (a week? a month? a year?) will not be negligible!

In this RFC, I've picked apart Bob Bressler's RFC 487, mostly because of its confusion of several distinct (although related) issues, and the implementation assumptions it contains which conflict with (or badly bend out of shape) mechanisms and design philosophies existing on other systems (in particular, the system I am most familiar with, Multics) [4]. The applicability of the discussions in this RFC, I think goes beyond that: We've got to acknowledge that it's difficult to propose Network-wide mechanisms for providing desirable services without building in assumptions about how they are to be implemented. We're at a point where we're asking for fairly sophisticated services, and proposing correspondingly sophisticated mechanisms. It's time to begin talking about how various systems accomplish such things as user authentication, access control, and so on, so that we can all gain a clearer understanding of such issues, and be able to propose mechanisms with fewer implementation assumptions built into them.

END NOTES:

[1] On some systems, there is a one-to-one correspondence between principals and accounts.

[2] It should be noted that systems which choose implementation 2) may require a user authentication sequence (USER, PASS, and possibly ACCT commands) before permitting any file transfers, as explicitly stated on page 17 of RFC 354 (NIC 10596), and page 20 of RFC 4554 (NIC 14333). This authentication sequence would be required to ascertain the principal identifier to be associated with the newly-spawned File Transfer Server process; the process is not allowed to proceed until its principal identifier has been established.

[3] Note that there are at least two scenarios for accomplishing the transfer Bressler desires: Either I "push" the file, using my "User FTP" and his system's "FTP Server", or he "pulls" the file, using his "User FTP" and my system's "FTP Server". Bob chose the first scenario; it can be argued that, since it is Bob who wanted the file transferred, the second scenario is the more appropriate one. A forthcoming RFC by Mike Padlipsky expands on these points, as well as an entirely different alternative.

[4] Padlipsky keeps insisting that I've also shown the superiority of implementation 2) of the File Transfer Server (described above), but I resist that conclusion. Those interested may want to look at his Unified User-Level Protocol specification, which is based on a similar premise.