How To Write a Setuid Program

Matt Bishop

Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA 94035

EXTENDED ABSTRACT

A typical problem in systems programming is often posed as a problem of keeping records [ALEP71]. Suppose someone has written a program and wishes to keep a record of its use. This file, which we shall call the *history file*, must be writable by the program (so it can be kept up to date), but not by anyone else (so that the entries in it are accurate.) UNIX† solves this problem by providing two sets of identifications for processes. The first set, called the *real* user identification and group identification (or UID and GID, respectively), indicate the real user of the process. The second set, called the *effective* UID and GID, indicate what rights the process has, which may be, and often are, different from the real UID and GID. The protection mask of the file which, when executed, produces the process contains a bit which is called the *setuid* bit. (There is another such bit for the effective GID.) If that bit is not set, the effective UID of the process will be that of the person executing the file; but if the setuid bit is set (so the program runs in *setuid mode*), the effective UID will be that of the owner of the file, not that of the person executing the file. In either case, the real UID and GID are those of the person executing the file. So if only the owner of the history file (who is the user with the same UID as the file) can write on it, the setuid bit of the file containing the program is turned on, and the UIDs of this file and the history file are the same, then when someone runs the program, that process can write into the history file.

These programs are called *setuid programs*, and exist to allow ordinary users to perform functions which they could not perform otherwise. Without them, many UNIX systems would be quite unusable. An example of a setuid program performing an essential function is a program which lists the active processes on a system with protected memory. Since memory is protected, normally only the privileged user *root* could scan memory to list these processes. However, this would prevent other users from keeping track of their jobs. As with the history file, the solution is to use a setuid program, with *root* privileges, to read memory and list the active processes.

Setuid programs introduce many security problems [TRUS80]; many of these can be dealt with by programming very carefully. The reader should bear in mind that on some systems, the mere existence of a setuid program introduces security holes; however, it is possible to eliminate the obvious ones.

By following some simple rules, programmers can decrease the danger of a setuid program being able to compromise system safety. These rules are:

†UNIX is a Trademark of Bell Laboratories.

1. Be as restrictive as possible in choosing the UID and GID of the setuid program.
2. Do not write interpreted setuid or setgid scripts.
3. Do not use *creat*(2) for locking.
4. Catch all signals.
5. Check data for consistency.
6. Take extreme care when recovering from errors.
7. Close all but necessary file descriptors before calling *exec*(2).
8. Reset effective UIDs and GIDs before calling. *exec*(2).
9. Check the environment of the process.
10. Be careful with I/O operations.

Setuid programs explicitly violate the protection scheme designed into UNIX. On systems where security is not a problem, this is a blessing, since it enables many things to be done easily that would otherwise be very difficult; but on systems where security is a problem, these programs also pose very real threats. Unfortunately, they are very necessary; so, the designers and implementors of setuid programs should take great care when writing them.

Acknowledgements: Thanks to Bob Brown, Peter Denning, George Gobel, Chris Kent, Rich Kulawiec, Dawn Maneval, and Kirk Smith, who made many constructive suggestions.

References

[ALEP71] Aleph-Null, "Computer Recreations," *Software — Practise and Experience* **1**(2) pp. 201 — 204 (April — June 1971)

[TRUS80] Truscott, Tom and Ellis, James, "On the Correctness of Set-User-ID Programs," Department of Computer Science, Duke University (unpublished)