

Security and Log Structured File Systems*

Tage Stabell-Kulø
Department of Computer Science
University of Tromsø
Norway
E-mail: *tage@acm.org*

January 9, 1997

Abstract

A log structured file system (LFS) obtains its performance by refraining from seeking for blocks on the disk. Instead, new data is written into new blocks which is placed at the end of the log. This incurs a security issue by leaving data on disk that was supposed to be overwritten.

To us, this shows how difficult it is for users concerned with their privacy, not to be lead astray. It also emphasizes how wide the gap is between an encryption channel and a user.

Introduction

A log structured file system (LFS) [4] obtains its performance by lazy garbage collection and by avoiding seeking for disk blocks. The performance gain can be substantial. The functionality of reclaiming old disk blocks is performed when needed, preferably at night. The key issue is that when data is written to a file it is always written to a physical location on the disk that optimizes the performance of the file system. Most file systems, among them are notably the Unix FFS [3] and MS-DOS, will reuse the physical blocks originally assigned to a file. That is, changing the data in the *file* will also change the data written to the storage medium (overwriting the old data).

One of the main objectives of a file system is to hide such details of physical storage, and users are not (in general) concerned with which physical disk block that actually was used to store data. However, this is not always the case.

If a file contains private data, it is well known that deleting the name of the file does not remove the contents of the file. On some systems, MS-DOS in particular, files can simply be “undeleted” and the entire contents recovered. On more sophisticated systems, such as Unix, this is not the case and more effort is required in order to recover data from a deleted file. In both systems the chances of success decreases with time, as measured in operations that modify the file system; the details are left out for brevity.

All this is well known, and quite some effort has gone into designing software helping users to (try to) discard their private data. In particular, it is also known that overwriting data with 0's (zero) is insufficient in systems that provide compression as a means to increase the effective size

*Printed in ACM Operating Systems Review, Vol. 31, No. 2, April 1997, pp. 9–10.

of disks, see, for example [1]. This is so since a series of 0's will usually compress better than the original data it was supposed to overwrite. The new version of the file is thus smaller than the old one, and (the end of) the original data is left unaltered on the storage medium. Subsystems such as compression are usually designed with great care and with compatibility in mind. That is, their presence can not reliably be detected by applications. Hence, the outmost care must always be taken. At this end, programs such as PGP¹ writes random data when overwriting, based on the fact that random data is not (significantly) compressible. To grasp the popularity of PGP one should notice that the "global" PGP key-ring² as of December 17th, 1996, contained no less than 34.838 public keys.

In a system with LFS, it is not possible to overwrite data in a file, since new disk blocks are always allocated. Although no file system API guarantees that the same physical disk blocks will be used, on most systems this can be assumed. On LFS, the opposite is true.

In other words, from a situation where the user could quite safely assume that the original data would be erased, the assumption on LFS is that the data almost certainly remains on disk (although outside the file system proper). This change is important, but can not be noticed by users that do not possess considerable insight into the workings of their system. Furthermore, this change is not detectable by tools the user might rely on.

Conclusions

A change in systems software can significantly alter the usefulness of users' tools to protect their privacy. In this paper we have shown that users on systems with LFS are particular vulnerable since they are unable to remove data from the file system.

In our view, the essence of this observation is related to what constitutes trust. Trust is needed to bridge the gap between the channel represented by some (secret) encryption key and a human that supposedly controls it [2].

The point we have made emphasizes that this gap is an abyss.

References

- [1] BURROWS, M., JERIAN, C., LAMPSON, B., AND MANN, T. On-line data compression in a log-structured file system. In *Proceedings of 5th ASPLOS* (1992), pp. 2–9.
- [2] LAMPSON, B., ABADI, M., BURROWS, M., AND WOBBER, E. Authentication in distributed systems: theory and practice. *ACM Transactions on Computer Systems* 10, 4 (Nov. 1992), 265–310. Also available as a technical report from <ftp://gatekeeper.dec.com/pub/DEC/SRC/research-reports/SRC-083.ps.gz>. A preliminary version of this paper appeared in the Proceedings of the 13th SOSP (1991).
- [3] MCKUSICK, M. K., JOY, W. N., LEFFLER, S. J., AND FABRY, R. S. A fast file system for unix. *ACM Transactions on Computer Systems* 2, 3 (Aug. 1984), 181–97.
- [4] ROSENBLUM, M., AND OUSTERHOUT, J. K. The design and implementation of a log-structured file system. In *Proceedings of 13th SOSP* (Oct. 1991), pp. 1–15.

¹For information about PGP, please refer to <http://www.pgp.com/> or, for non-US users, <http://www.ifi.uio.no/pgp/>.

²For details about the "official key-ring" effort, please refer to <http://www.pgp.net/pgpnet>.