

Project Proposal: Secure Isolated Copy-on-Write File System

Fareha Shafique
September 26, 2006

Introduction

When several users are working with the same files, it is helpful to isolate the changes made by each user since some users maybe malicious. At the same time, different users might be allowed to modify the files according to different permissions. For example, consider a web server where a user has hosted his photo gallery. At the end of a session several modifications may have been made, some of which should be discarded, however, the user will most likely want to save the picture files that were uploaded. One way to achieve this is to direct all changes (as they are made) to one 'isolated, secure area' and then allow the user to selectively commit them. This is similar to Concurrent Versions System (CVS) with the difference that CVS only allows checkouts of the entire directory tree rather than only those files that are modified. Furthermore, to ensure security and privacy, users may have different permissions that restrain their ability to commit. At the moment, several web applications provide such services as part of the application. I believe that moving this into the file system layer can provide better security and web management, as well as uses in other areas in the future such as mobility and migration.

Proposed Solution

I propose developing a user-level copy-on-write file system using FUSE (File system in User Space) [1]. The system will consist of a base layer and a copy-on-write layer. It will allow a user to commit changes to the base layer (which may itself have changed resulting in conflicts) according to a set of policies to be developed as a future project.

A copy-on-write (CoW) file system allows reads to execute normally, but creates a copy of the file in its own space when any modification is done. Thereafter, all reads and writes to this file are redirected to the new copy. In this way, the underlying file system remains unchanged by all the modifications and it is possible to either revert to it by discarding everything in the CoW file system or to merge the changes.

FUSE allows a user (developer of a file system) to mount a directory onto the new file system. It then works by intercepting file system related system calls at the virtual file system (VFS) layer, and redirecting execution to the user level code that implements the new file system. Any file system call in this code will not be intercepted, but will execute as normal. Therefore, implementation of the proposed file system involves writing wrappers, for those calls that are supported by FUSE, to carry out the copy-on-write and isolation functionality.

FUSE consists of a kernel module and a library that communicate through a special file descriptor obtained by opening `‘/dev/fuse’`. This file can be opened multiple times, and the obtained file descriptor is passed to the mount system call in order to relate this file descriptor to the mounted file system. FUSE comes with several example file systems.

Related work

The most closely related work is ‘One-way Isolation: An Effective Approach for Realizing Safe Execution Environments’ [2]. Sekar et al. have developed an isolated file system which also intercepts file system calls at the VFS layer. Their model views a file system as a directed acyclic graph (DAG) and consists of a base file system layer and an isolated file system (IFS) layer, the combined view of which gives the current state of files and directories. They ensure that any application running in an isolated file system cannot make changes to the main system and hence cannot harm the host in any way. Their system also allows commits. The main difference between their work and the proposed work here is the intended use of the isolated file system. While they have used it to disallow an application to effect the host file system, the intended use of the proposed project is to allow web management, mandatory access policies for a file system and possible migration too.

Another file system that provides copy on write functionality is ‘ext3cow’ [3]. However, its implementation is embedded in the ext3 implementation and is intended as a replacement to it. My system builds on top of the existing file system and since it intercepts system calls at the VFS layer it is compatible with many different underlying file systems. Furthermore, ext3cow is used to provide a form of versioning where different versions of the same file system can be accessed according to an epoch number and thus it does not provide the isolation the proposed system aims to provide.

Several other versioning file systems exist such as ‘Wayback’ [4], which also uses FUSE to develop a user-level file system that logs all changes to the base system and makes them visible to the mounted system but also allows the user to revert to any older version of any file. They appear to keep separate copies of files whereas, actually, they just keep logs and provide the user a merged view of the logs and the base files. This neither provides isolation nor copy-on-write and provides many versions of the same file, and is, hence very different from the proposed system.

As mentioned before, one proposed use of the file system is to provide mandatory access control policies to the files. SELinux [5] integrates security policies into the Linux operating system to offer such mandatory access control. However, the policies for SELinux are very complicated and I am hoping that using a secure isolated file system will allow for simpler policies.

References

- [1] FUSE, <http://fuse.sourceforge.net/>
- [2] W. Sun, Z. Liang, R. Sekar, V.N. Venkatakrisnan: One Way Isolation: An Effective Approach for Realizing Safe Execution Environments. In *Proceeding of Network and Distributed System Security Symposium, 2005*.
- [3] Z.N.J. Peterson, R.C. Burns: Ext3cow: The Design, Implementation, and Analysis of Metadata for a Time-Shifting File System. *Technical Report HSSL-2003-03*.
- [4] B. Cornell, P.A. Dinda, F.E. Bustamante: Wayback: A User-level Versioning File System for Linux. In *Proceedings of USENIX Annual Technical Conference, 2004*.
- [5] P. Loscocco, S. Smalley: Integrating Flexible Support for Security Policies into the Linux Operating System. *NSA Technical Report, 2001 and Proceeding of the USENIX Annual Technical Conference, 2001*.