Frozen Files

Frank Kardel
Friedrich Alexander Universität Erlangen-Nürnberg
IMMD $\overline{\text{IV}}$ • Martensstrasse 1 • D-8520 Erlangen • (West-) Germany

1. Problem

At the university of Erlangen-Nürnberg we are running a moderately large network of computers from different vendors running various variants of the Unix[1] operating system. These machines are administered by students and faculty staff of seven faculties. Unfortunately most machines are basically insecure since vendors not always deliver the machines correctly configured in respect to security. Furthermore Unix was not designed to be extremely secure in the first place being a software development system.
The faculty staff cannot secure all machines, so only some relevant machines such as file servers and source machines are taken care off. The remainder, workstations and machines for educational purposes, are basically kept running in a fairly good condition.
The main problem is that it is currently virtually impossible to insure that only trusted users can become the *Superuser* “root”. Since “root” is allowed to change anything in a Unix system there is little protection against destruction of files and other manipulations that will for example allow easier re-entry for intruders.
For this reason we needed a mechanism to limit the damage that can be inflicted by “root”. The best thing would be a protection scheme that is easy to implement and does not depend on simple mechanisms as user identification numbers (Uid’s). Another requirement was that the protection scheme should be compatible with the standard Unix protection scheme in order to allow almost every program to run without change.

1. Unix is a trademark of AT&T Bell Laboratories

2. Frozen Files

Frozen Files is an extension of the standard Unix permission checking system. Files have two mode fields instead of one for determining the rights of a process in respect to a file. Access permissions for a process are checked against the so called “effective mode” which is derived from the “public” and the “private” access mode of the file.
Files can exist in two forms: **normal** or **frozen**. A normal file has only the “public” access mode field which is checked with standard Unix access semantics (normal permissions checking, permissions are ignored for “root” - as usual). A frozen file possesses both the “public” and “private” access modes.
The process of adding the “private” access mode to a file is called freezing. Freezing is done by adding a password to a file. Certain file system operation are forbidden for a frozen file.
These are:

link, unlink, chmod, chown, rename

In order to regain full access to the file the password the file is frozen with must be added to the credentials of the process wishing to manipulate the file. The checking of access rights is changed for frozen files. Processes not in possession of the correct file password are checked against the “public” access mode. Since “root” can change its user id to any value, it can gain access to files where at least one bit is set in the user, group, others field of the “public” mode. So if there is no “write” bit set in any of the user, group, others fields then write access is denied for “root”. This semantic allows to effectively deny certain types or even all access permissions for “root”. If the process has the correct password for the file its access permissions are checked against the logical OR of the “public” and the “private” access mode. Furthermore, the link, unlink, chmod, rename operations are allowed again if the effective access mode permits these operations.

So far, a two level permission checking mechanism has been shown, where “root” cannot break the rules without knowing the correct file passwords. There still remains one more problem with Unix file system semantics.

2.1 Frozen Special Device Files

Special device files allow “root” to do anything since they allow direct access to device drivers and thus raw disc access and on many systems also access to kernel data structures. In order to protect special device files it is not sufficient to protect a single device file entry in the file system because “root” could still create new unprotected special device entries and thus circumvent the protection mechanism. So it is important to protect the device drivers instead of the device files. The frozen file mechanism insures correctly installed special device files by requiring that device files are frozen for protected drivers. This requirement is not enough since “root” could still create a new special device file and freeze it. For this reason another requirement was added: Special device files for protected drivers must be frozen with the same password the system root mount point (“/”) is frozen with. The distinction whether a driver is protected or not is made via an additional flag in the *cdevsw* and *bdevsw* structure. There is a need for devices not to be protected by the frozen file mechanism, terminal drivers for example.

2.2 New Rights for Frozen Files

In order to manipulate frozen files a knowledge of a password is required. These passwords are added by a new system call to the process’ credentials in encrypted form. It is possible to add several passwords to this list so one can manipulate several differently frozen files. For every password a flag can be set that defines whether the password should be deleted on the execution of the “exec” system call or not.

The s-bit mechanism of Unix finds its counterpart in the p-bit mechanism for frozen files. A frozen file can have the p-bit set. Upon execution of this file the password the file is frozen with is added to the process’ password list. This mechanism makes it possible to give programs and scripts additional rights in respect to frozen files.

The p-bit mechanism supports accumulative rights in contrary to the one level s-bit mechanism. Passwords added via p-bits will normally be destroyed on execution of the “exec” system call, unless specified otherwise. This allows interpreters to run with a higher privilege level without the risk of passing password rights to untrusted programs. Another advantage of frozen files is that password rights cannot easily be stolen (saved) by creating p-bit files since the original unencrypted password is required to create the p-bit file.

3. Frozen File Protection Issues

Frozen Files can be used in several ways to increase security. The major advantage is that they are resistent to ordinary “root” users who do not have the correct password. This property allows the installation of programs and configuration files with the advantage, that a “root” user cannot change or move these files, although otherwise having full control over the system.

3.1 Controlled Configurations

It would be advisable to use the frozen file mechanism for protecting the /etc/rc group of files and thus separate the system configuration aspects from administrative “root” user actions.

3.2 Controlled Access

Frozen Files can also limit the privileges given out to certain programs to that, that is needed for the required action.

3.2.1 A Limited Rights Example

A *passwd* program could be frozen with the same password the passwd-file is frozen with. The *passwd* program could then have a p-bit and the private mode of the frozen file is set to be writable by all users, while the public mode of the passwd file is set to be read only. This setup would make the *passwd* program to be the only entity being allowed to change the password- file (except for the Person knowing the passwd file password). The advantage is that not even “root” could not break the rules. So it is possible to build different administrative domains. The danger of rights leaking out of a program can be controlled by the *delete on exec* flag that is associated with each password.
Freezing central system files can also lead improved resistance against worms and viruses.

3.2.2 Frozen Files for Privileges

Due to the possibility of carrying multiple passwords, the Frozen File mechanism is also suitable to build a system of privileges, like the compilation privilege, the system source privilege and alike by using a modified *login* program to set the passwords in the credential structure.

3.2.3 Credential Passwords

Currently a simple mechanism for storing the passwords in the credential structure is used. The password list consists of pairs of facility mask and password value. The facility mask defines the functions the password value can be used for and whether it is to be deleted on the execution of the “exec” system call. Right now only one facility is defined: Frozen Files. The password list concept a a simple Capability system which can be used for more than just verifying file access permissions. The protection of certain system calls comes into mind and the possibility to encode more rights into the passwords by cryptographic means.

3.3 Device Access Restriction

The possibility of bypassing the protection mechanisms is greatly reduced by having frozen devices. This allows modification of disk data structures and kernel data structures only to a very limited number of users. The protected devices still cannot control the physical access of disks and tape and thus cannot prevent the booting of a suitably modified kernel or direct disk block access, but this mechanism can be hardened by using cryptographic techniques at the *blockio* level.
Though not being able to prevent physical access Frozen Files can be used to harden a system against attacks from the running system, such as patching the kernel or installing a new bootloader.

3.4 Controlled Boot

Since the root filesystem (“/”) can be frozen (and must be frozen for protected devices to work) a modified bootloader must be installed on the root filesystem. This bootloader can check the integrity of the operating system image to be loaded by requiring that the operating system is frozen the same way the root directory is frozen with. This requirement insures, that only correctly frozen kernels can be booted. In order to allow booting of alternate kernels a boot can also be permitted, when the user enters the password of the root directory. This mechanism can be circumvented only, if physical device access is present or when the system monitor allows memory modification. Some vendors can deliver “secure” monitor ROMs that will forbid this kind of modifications.

4. Implementation Aspects

Frozen Files do not require much changes. Frozen Files where implemented in SunOS 4.0.3. There have been few changes to some parts of the kernel and some user level programs.

4.1 Complexity

All changes to the kernel where made during a 2 week operating systems project by 12 students who have never done any kernel work before. This shows that required changes are acceptable for the achieved results.

4.2 Frozen File Implementation

The frozen files are implemented as a new file type in the ufs file system. The SunOS vnode-operations have been extended by a new operation that allows the frozen file manipulations. The additional data needed for the frozen files was stored in some unused fields of the disk inode structure. The stat system call also passes the additional data in unused fields of the stat structure. The mode field of the stat structure is set to the effective mode of the file. This allows all programs that have no knowledge about frozen files to run without any compatibility problems.
A few user level program have been extended to allow easy manipulation of the frozen files.

Kernel changes:

- ufs files
  new access mode check, special device support
- new system call
  implementation of password management routines, freeze and thaw operations
- os files
  p-bit semantic for exec system call
  support for new credential structure
- boot files
  new bootloader for frozen files

Changes in user level programs:

- chmod
  allow changes of private mode and setting of p-bits
- csh/sh
  support password addition and deletion
- ls
  display public and private mode of a file (new option)
- fsck
  support frozen files
- find
  allow search for frozen files and specific private modes

5. Summary

Frozen files are only a mechanism and it is yet to be determined how usable they are and whether they will contribute to security. One major advantage is the simple implementation and good compatibility to standard Unix system call interface. Frozen files are currently implemented on a Sun 3 Workstation[2] under SunOS 4.0.3 at the university of Erlangen-Nürnberg.

2. SunOS, Sun Workstation ®, as well as the word “Sun” followed by a numerical suffix, are trademarks of Sun Microsystems, Incorporated.