Shadowfs

A framework for LD_PRELOAD

filesystem wrappers

Clifford Wolf - www.clifford.at
Introduction
Overview

- shadowfs is a small framework for writing filesystem wrapper LD_PRELOAD libraries.

- three such libraries included in shadowfs:
  - liblogfs - a filesystem operations logger
  - libcowsfs - a copy-on-write translucent filesystem
  - libmmfs (under construction) - a wrapper for simulating root permissions
ROCK Linux

- shadowfs has its roots in the ROCK Linux projects
- liblogfs will replace the currently used flwrapper.so
- libcowfs is used in the live CD target
- libmmfs will be used for some advanced build methods
Building shadowfs

- Simply running "make" and "make install" should do the job

- Maybe you need to adapt some settings in config.h:
  - DEBUG, DEBUG_386 - Enable internal debugging.
  - DOPEN_LIBC - Try switching this option when you encounter troubles.
  - GLIBC_IS UGLY - Set this to 0 if you are not using glibc (e.g. for dietlibc based systems).

- Older binutils (i.e. the binutils debian package) screw up at "objcopy --keep-global-symbols=symbols.txt".
Overview

- This library monitors all file operations.

- It is configured using environment variables:
  - `LOGFS_ROLOG` - the log file for read/execute operations
  - `LOGFS_RWLOG` - the log file for write operations

- The logfiles must exist already when liblogfs is started.

- The logfiles include the command tree and function which issued the operation the the affected filename.

- The command tree is terminated at the PID stored in `LOGFS_BASEPID`. This environment variable is set automatically by the first process.
Usage scenarios

- Debugging huge applications (faster than strace/ltrace)

- Automatically create file lists for "make install".

- Profiling which files (and packages) are used while performing a task (e.g. when doing package selections for small distributions).
Overview

- This library creates a virtual 'translucent' filesystem.

- The mechanism uses a 'read-write' and a 'read-only' directory.

- On default every subdir of the 'read-only' master is symlinked to the 'read-write' directory.

- Whenever a write on the 'read-write' directory is done, the symlink will be replaced with a copy.

- The directory paths are configured using the `COWFS_RO` and `COWFS_RW` environment variables.
The symlink approach used here has many advantages:

- It is very clear what libcowfs is doing and how
- Changes in the 'read-only' directory are visible and cause no harm (important for NFS root environments).
- Backing up changes or reverting to the original state is very easy.
- Statically linked applications can still access the filesystem and even write after a previously done copy-on-write.
The CWD hack

- I is possible that a write operation must result in a chdir:
  - Process X is in a deep subdirectory level which has not been COWed yet.
  - The process tries to write to a file in this directory.
  - The file gets COWed. In order to do that, the directory is created on the read-write filesystem and all files are symlinked. The target file of the operation is copied.
  - Now process X is in the wrong directory.

- In order to deal with this problem, libcowfs is changing the current working directory in such cases.

- It also can handle such a situation if a child process has triggered the copy-on-write, but only when wait() is used to wait for the child process.
Usage scenarios

- Building Live-CDs

- Building Root-NFS environments

- Testing with ability to ‘roll back’ to the original state.
libmmfs

Overview

Usage scenarios

NFS Root (cowfs)

Live CDs (cowfs)

URLs and References
Overview

- libmmfs is the 'megamaniac' filesystem. It is not finished yet.
- It allows a normal user to virtually change anything in the system.
- Changes are written to a copy-on-write directory.
- The copy-on-write data includes metadata such as userid and permissions.
- It is not as stable as libcowfs because the symlink mechanism is not possible here and so also read access must be rewritten.
Usage scenarios

- Building and testing software as normal users while making it look to the application as if it would be installed system-wide.

- For more advanced build and regression-test methods in ROCK Linux.
NFS Root (cowfs)
Overview

- An NFS server exports a full root filesystem and a minimalistic nfs boot environment.

- Both NFS exports are read-only. All local changes are written to the workstations RAM.

- The server config is the same for one of hundred workstations.

- Optionally DHCP and TFTP servers may be used to PXE-boot the workstations.

- For the ease of administration the exported root filesystem may be the distribution running on the NFS server.

- A script (nfsroot.sh) for such a setup is included in the shadowfs sources.
Using nfsroot.sh

- A simple "make nfsroot" creates an `nfsroot/` directory using `nfsroot.sh`.

- The files from a user-supplied `dot3/` directory are copied to `nfsroot/...` and are used by the workstations for various configurations.

- The `nfsroot/` directory must be used by the workstations as root filesystem.

- `/` from the NFS server is automatically mounted at `/mnt/cowfs_ro` on the workstations.

- `/home` is mounted read-write from the NFS server and is not COWed.
Adapting nfsroot.sh

- It might be necessary to make some changes to `nfsroot.sh` to fit your specific needs.

- The script is pretty straightforward.

- Re-running "make nfsroot" is possible without causing troubles with already connected workstations.

- Updating packages in the exported root filesystem is also possible without much trouble.
Live CDs (cowfs)
Implementing Live CDs

- The 'usual' root filesystem data is moved to /mnt/cowfs_ro.

- The /mnt/cowfs_rw directory just has symlinks to the entries in /mnt/cowfs_ro.

- The root directory just has symlinks to /mnt/cowfs_rw.

- libcowfs.so is loaded from /etc/ld.so.preload.

- Somewhere in the boot process (e.g. where usually / is mounted read-write), a tmpfs is created with the same content as /mnt/cowfs_rw and is moved (using mount --move) over /mnt/cowfs_rw.

- This is very similar to what the init script created by nfsroot.sh does.
URLs and References
The shadowfs sources:
http://svn.clifford.at/shadowfs/trunk/

ROCK Linux:
http://www.rocklinux.org/

Clifford Wolf:
http://www.clifford.at/

LINBIT Information Technologies
http://www.linbit.com/