Fun with glibc's runtime linker

Marvin Blauth

August 30, 2016

The rtld has parameters!

- The rtld can be used as executable, e.g. /lib64/ld-2.23.so
- For the dynamically linked rtld environment variables can be used, e.g. LD_LIBRARY_PATH

Interesting rtld environment variables

- LD_BIND_NOW – Force resolving of all symbols on startup
- LD_PRELOAD – Preload symbols from selected library
- LD_DEBUG – Print debug information for linker activity
- LD_AUDIT – Register an audit library to be invoked on linker activity
- There are many more options, check out **man ld.so**

LD_AUDIT

- glibc's implementation was based on the auditing interface present on Solaris
- The auditing interface enables you to register symbols to be called on almost any step of the linker activity
- Example: When searching for a symbol in a shared library lib.so, the symbol **la_objsearch** will be called with "lib.so" as parameter.

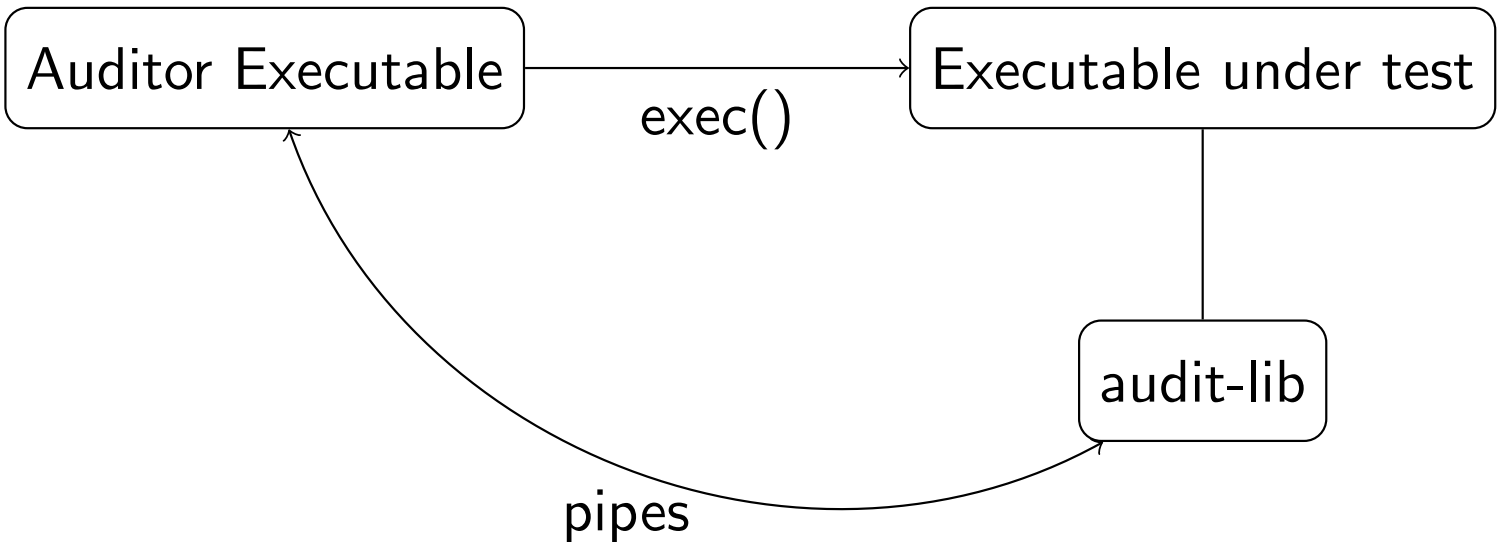
Using LD_AUDIT for fun and profit

- The auditing interface can not just monitor the linker activity, but also reject it
- This can be used for performing any sort of actual audit on the linker activity, for instance
  - OSS license compatibility
  - Library file checksum
  - Installation source of library

Using LD_AUDIT for checking the installation source 1/2

- Imagine you want your software to only use symbols from the system repositories
- LD_AUDIT in combination with libhawkey/libhif/libdnf lets you very easily do that on Fedora

Using LD_AUDIT for checking the installation source 2/2



Sources

- `https://github.com/mblauth/runtime-linker-audit`