NAME

user-union − User-space union mount, implemented using LD_PRELOAD.

SYNOPSIS

user-union [*OPTIONS*]... [[--] *command...*]

DESCRIPTION

This command implements "union mounts" without requiring special privileges. In a union mount, writes to one directory (the "underlay") appear to happen normally, but any changes are actually performed in a separate parallel directory (the "overlay") instead. Contents in the overlay take precedence over the contents in the underlay, so once a file is "written" in the underlay, reading it back from the underlay shows the "new" contents. The result is a tree of directories that appear to be modifiable, but modifications are occurring somewhere else.

Union mounts can be used for many tasks. For example, they can be used to make read-only media appear to be read-write ("user-union -u /mnt/cdrom" will appear to make /mnt/cdrom writeable). They can also redirect software installation processes so that files that appear to be written in one place are instead written elsewhere else (automating DESTDIR). Unlike many union mount systems, user-union can be used by users without any special privileges.

If you *do* have root privileges, use another union mount system instead of user-union. User-union uses LD_PRELOAD to implement union mounts, and thus has many inherent limitations. However, if you need a way to do union mounts *without* privileges, this may be the tool you are looking for.

A "branch" is a statement that sets an underlay/overlay pair, or a statement that some directory is to be ignored (not redirected), and can be set using "-a", "-i", and "-m" (see below). If there are no options to set branches, then a default set of branches is used. This default is as follows, where *OV* is the default overlay directory (set by -o, otherwise user-union creates a temporary directory) and *UN* is the default underlay directory (set by -u, otherwise "/"):

-a *OV UN* -i /tmp -i /var/tmp -i /home -i /Users

When accessing a particular directory, the longest-matching directory name is used; if it matches more than one, the first one is used.

After this is set, the *command* is run. If no command is given, the $SHELL is executed; if $SHELL is not set, it is considered to be /bin/sh. If you run the shell, exit the shell to end the overlaying set up by user-union (this is typically done using control-D, control-Z ENTER, or exit 0 ENTER).

OPTIONS

−a *OV UN* Add an overlay, where overlay *OV* overlays underlay *UN*. Attempted writes to "underlay" will instead write to "overlay", and any contents of "overlay" will override those in "underlay".

-i *DIR* Ignore *DIR* - do not overlay it, even if it’s within a directory that is overlaid.

-m *UN* Simulated mount. Create a temporary overlay directory TEMPORARY, print that directory name TEMPORARY as the first output line, and then do **-a** *TEMPORARY UN*. If you want to create a "writable" directory region and don’t care what overlay directory is used, use this.

-n No-operation (dry run). Print the resulting USER_UNION using "od -c" and exit. This is intended for testing purposes.

-o *OV* Set the default overlay directory to *OV* which is used when there are no other commands determining what to overlay; see the description.

-t Test mode. Use current directory’s "user-union.so" library, instead of searching the usual library directories.

-u *UN* Set the default underlay directory to *UN* which is used when there are no other commands determining what to overlay; see the description.

IMPLEMENTATION APPROACH

This tool is implemented using LD_PRELOAD, which has various implications. For example, any privileged program (like su) will *not* be redirected.

This tool can be combined with other libraries that use LD_PRELOAD, though whether or not this works depends on the other libraries. The user-union tool must modify LD_PRELOAD to work; it will add the user-union shared library to the beginning or end of the LD_PRELOAD if a space is on that side, if you need to force its location.

The branch settings for user unions are stored in the environment variable USER_UNION. If the env variable USER_UNION is already set, then we may have user-union being used inside user-union, aka "stacking". In this case, USER_UNION will be prepended with the value described above.

The environment variable USER_UNION_SO, if set, is the name of the .so library to be run. Otherwise, 'user-union.so' is used (in the current directory if -t is set). If it can't load the shared library, you will see error messages like: ERROR: ld.so: object 'user-union.so' from LD_PRELOAD cannot be preloaded: ignored.

The underlay directory has a hidden directory ".user-union" created, and this directory is silently added to as an "ignore" (aka non-union) branch. This hidden directory is used to store housekeeping information and temporary data.

EXAMPLES

The following command will start up an interactive session (with $SHELL) in which /mnt/cdrom *appears* to be writable:

```
user-union -m /mnt/cdrom
```

Here, writes to anywhere below "/" are overlaid with the directory /tmp/redir, except for /home, /tmp, and /var/tmp, creating the impression that the user can write on many places such as /usr/bin:

```
user-union -a /tmp/redir / -i /home -i /tmp -i /var/tmp
```

LIMITATIONS

This is an early release. The current implementation has lots of limitations that could be fixed in future versions. Currently it just focuses on intercepting open(), fopen(), chdir(), unlink(), rename(), and a few other functions like it. This means that the current implementation:

* doesn't implement opendir() / readdir() and friends, so lists of files in an underlay won't list files created in the overlay, and will list files that were removed. This isn't terribly hard, it just requires more code to do it.
* doesn't intercept some other functions that it *should* intercept. In particular, it should be intercepting any function call that takes a filename. Functions it doesn't intercept, but probably should, include the standard functions execl(), execlp, execle(), and getcwd(), some *at functions, and the new open by handle system calls of Linux like open_by_handle_at (see https://lwn.net/Articles/432757/). Anyone who wants to help should look at the list of functions in the C library (including system calls) that take function parameters, but aren't yet wrapped.
* doesn't implement various edge cases of some functions. For example, when it executes a file without a '/' in the name, it doesn't currently implement the exactly-correct search algorithm. It also doesn't

implement the *at functions exactly correctly; it basically ignores the extra "at" parameter. This not a limitation of the approach, it's just that implementing some functions precisely requires code that hasn't been written yet. In many cases this doesn't matter.

* doesn't simulate many access control (privilege) checks. As a result, it currently acts more or less like how the system acts for the root user when it can manage to simulate the operation at all.

More seriously, because it uses LD_PRELOAD, it is subject to many fundamental limitations:

* Any privileged program (like "su") will not be redirected.
* A program that is statically linked can't be redirected by any LD_PRELOAD based tool, including this one. On most Linux-based systems this isn't a problem, as very few programs are statically linked to low-level libraries like the C library. However, on some platforms (especially embedded systems) a few important basic commands are statically linked (such as cp, ln, and so on). If this is your situation, you might be able to use the auto-destdir package instead or in addition to user-union. See run-redir(1) in the auto-destdir package, and the run-redir-union(1) "-a" option, for more about doing this.
* There will always be calls that it doesn't redirect, so there will always be ways for important information to be revealed.
* The C library's internal calls often cannot be overridden, depending on the specific implementation. In particular, the GNU C library's default installation makes it impossible to redirect "internal" calls inside the GNU C library. The user-union package compensates for this by overriding many additional functions, e.g., it overrides fopen() as well as open(). Nevertheless, such systems the "union mount" abstraction is especially leaky. On the GNU C library could resolve this by recompiling the GNU C library to enable redirection (using the "--disable-hidden-plt" option) and use it. For many users, recompiling their C library just to do this is not a practical solution. Even if your C library allows redirections of its internal functions (e.g., you compiled the GNU C library with "--disable-hidden-plt"), this kind of tool will always be a leaky abstraction.
* There is the possibility that a change in the underlying C library will end up causing serious problems in this library. The fakeroot implementation had this problem and gave up. The author is aware of these issues, and has decided to do this anyway :-). In particular, user-union implements an "override prefix" and has been written in a way to resist this kind of problem.
* Changing environment variables used by user-union could interfere with it. In particular, changing LD_PRELOAD in a way that removes the shared object user-union.so will disable user-union. Changing the USER_UNION environment variable, which stores how to redirect information, will also affect user-union, though in that case presumably that's what you wanted to do. The USER_UNION variable is only read on process startup, so any changes to the variable can only affect other processes started up by the process.

SECURITY CONSIDERATIONS

This program does not grant or require any special permissions. It will make it *appear* that programs have special privileges, but this is an illusion created by saving and retrieving information in other locations. Attempts to read files the user cannot read at all will still fail. Attempts to write to special files (e.g., block devices) that the user cannot write to will still fail as well. Attempts to write files into privileged places will only work, when they work, because the system is actually writing somewhere else. Running a setuid/setgid program disables the illusion, so user-union cannot subvert setuid/setgid programs on a correctly-configured system.

That said, this can become a security problem if a user uses it to fool a root user into doing something they shouldn't (e.g., typing their root password into something that captures it). Administrators with root privileges should continue to ensure they're talking to the real login program before typing in their password (this is known in the security field as having a "trusted path"). This would be true whether or not user-union existed.

If a system has a security mechanism that grants additional privileges when certain programs are run, then the system must disable or ignore LD_PRELOAD. Systems that implement setuid/setgid typically do that, but if there are other such mechanisms, those mechanisms must also disable or ignore LD_PRELOAD. If a system fails to do so, then it already has a vulnerability, whether or not user-union is installed. Again, user-union is not creating the vulnerability; a system that fails to disable or ignore LD_PRELOAD in these cases already has a vulnerability.

AUTHOR

David A. Wheeler

REPORTING BUGS

Report bugs to <dwheeler, at, dwheeler dot com> See http://www.dwheeler.com/user-union for more information.

COPYRIGHT

SEE ALSO

run-redir-union(1), run-redir(1), make-redir(1), fakeroot(1).