Firejail

Firejail (https://firejail.wordpress.com/) is an easy to use SUID sandbox program that reduces the risk of security breaches by restricting the running environment of untrusted applications using Linux namespaces, seccomp-bpf and Linux capabilities.

Related articles

Security

Contents

Installation

Install either `firejail (https://www.archlinux.org/packages/?name=firejail)`, `firejail-git (https://aur.archlinux.org/packages/firejail-git/)`[AUR] or the `firejail-apparmor (https://aur.archlinux.org/packages/firejail-apparmor/)`[AUR] package. A GUI application for use with Firejail is also available, `firetools (https://aur.archlinux.org/packages/firetools/)`[AUR].

Note: The `user_namespaces(7)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/user_namespaces.7) are available only in `linux` (https://www.archlinux.org/packages/?name=linux) (v4.14.5 or later) and `linux-hardened` (https://www.archlinux.org/packages/?name=linux-hardened). Impact on Firejail users is **deemed negligible** (https://github.com/netblue30/firejail/issues/1347)

Warning: While upstream is gradually adopting whitelists, (cf `/etc/firejail/firefox.profile`,) most of the supplied profiles still rely heavily on blacklists. This means that anything not explicitly forbidden by the profile will be accessible to the application. For example, if you have btrfs snapshots available in `/mnt/btrfs`, a jailed program may be forbidden from accessing `$HOME/.ssh`, but would still be able to access `/mnt/btrfs/@some-snapshot/$HOME/.ssh`. Make sure to audit your profiles, see **#Testing profiles**

Configuration

Most users will not require any custom configuration and can proceed to **#Usage**.

Firejail uses profiles to set the security protections for each of the applications executed inside of it - you can find the default profiles in `/etc/firejail/application.profile`. Should you require custom profiles for applications not included, or wish to modify the

defaults, you may place new rules or copies of the defaults in the `~/.config/firejail/` directory. You may have multiple custom profile files for a single application, and you may share the same profile file among several applications.

If firejail does not have a profile for a particular application, it uses its restrictive system-wide default profile. This can result in the application not functioning as desired, without first creating a custom, and less restrictive profile.

Refer to **firejail-profile(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/firejail-profile.5)**.

Usage

To execute an application using firejail's default protections for that application (the default profile), execute the following:

```
$ firejail <program name>
```

One-time additions to the default profile can be added as command line options (see the man page). For example, to execute *okular* with seccomp protection, execute the following:

```
$ firejail --seccomp okular
```

You may define multiple non-default profiles for a single program. Once you create your profile file, you can use it by executing:

```
$ firejail --profile=/absolute/path/to/profile <program name>
```

Using Firejail by default

To use Firejail by default for all applications for which it has profiles, run the *firecfg* tool as root.

```
# firecfg
```

This creates symbolic links in `/usr/local/bin` pointing to `/usr/bin/firejail`, for all programs for which firejail has default profiles. Once this is done, you only need to prefix a program with *firejail* if you want to run it with some custom security setting.

Please be aware that in order for this to work, `/usr/local/bin` must be before `/usr/bin/` in your `PATH`.

You can manually do this for individual applications by executing:

```
# ln -s /usr/bin/firejail /usr/local/bin/<program name>
```

Note:

- For a daemon, you will need to overwrite the systemd unit file for that daemon to call firejail, see **systemd#Editing provided units**.
- `firecfg` doesn't work with some cli shells such as: `tar`, `curl`, `wget`, `git` and `ssh` which need to be symlinked manually.
- Symbolic links to `gzip` and `xz` interfere with `makepkg`'s ability to preload `libfakeroot.so`. See **BBS#230913 (https://bbs.archlinux.org/viewtopic.php?id=230913)**.

Warning: Upstream provides profiles for `gpg` and `gpg-agent`. If gpg is symlinked with the supplied profile, pacman will be unable to update **archlinux-keyring (https://www.archlinux.org/packages/?name=archlinux-keyring)**.

Verifying Firejail is being used

```
$ firejail --list
```

Creating custom profiles

Whitelists and Blacklists

Blacklists are permissive:

- Permit everything not explicitly forbidden: `blacklist <location/file>`
- Permit file or location in any later blacklist: `noblacklist <location/file>`

Whitelists are restrictive:

- Forbid everything not explicitly permitted: `whitelist <location/file>`
- Forbid file or location in any later whitelist: `nowhitelist <location/file>`

Profile writing

The basic process is:

1. Copy the default profile (which uses blacklists) to your work folder and give it a unique name:
2. Change the line `include /etc/firejail/default.local` to `include /etc/firejail/ProfileName.local`
3. Gradually comment/uncomment the various options while checking at each stage that the application runs inside the new sandbox
4. Desirable options not available in the copied default profile can be found by consulting the manual
5. **Build a whitelist (https://firejail.wordpress.com/documentation-2/building-whitelisted-profiles/)** of permitted locations. For portability, it may be advisable to place at least some of this list it in a `.local` file
6. Test the profile for security holes, see **#Testing profiles**

7. Once satisfied, copy your new profile to either `/etc/firejail/` or `~/.config/firejail/`

You may find the following to be useful:

1. `firejail --debug $OtherOptions $PathToProfile $Program > $PathToOutputFile`
   Gives a detailed breakdown of the sandbox
2. `firejail --debug-caps` gives a list of caps supported by the current Firejail software build. This is useful when building a **caps whitelist (https://l3net.wordpress.com/2015/03/16/firejail-linux-capabilities-guide/)**.
3. `firejail --help` for a full list of `--debug` options
4. `firemon PID` monitors the running process. See `firemon --help` for details
5. **`checksec (https://www.archlinux.org/packages/?name=checksec)`** may also be useful in testing which standard security features are being used

Note:

- The idea is to be as restrictive as possible, while still maintaining usability. This may involve sacrificing potentially dangerous functionality and a change in cavalier work habits.
- By default, seccomp filters work on a blacklist (which can be found in the manual). It is possible to use `seccomp.keep` to build a custom whitelist of filters for an application. **[1] (https://firejail.wordpress.com/documentation-2/seccomp-guide/)**.

- The list of possible options for a firejail profile is extensive, and users should consult the firejail-profile(5) man page.

Persistent local customisation

The standard profile layout now includes the capability to make persistent local customisations through the inclusion of `.local` files. Basically, each officially supported profile contains the lines `include /etc/firejail/ProgramName.local` and `include /etc/firejail/globals.local`. Since the order of precedence is determined by which is read first, this makes for a very powerful way of making local customisations. For example, with reference **this firejail question (https://github.com/netblue30/firejail/issues/1510#issuecomment-326443650)**, to globally enable Apparmor and disable Internet connectivity, one could simply create/edit `/etc/firejail/globals.local` to include the lines

```
# enable Apparmor and disable Internet globally
net none
apparmor
```

Then, to allow, for example, "curl" to connect to the internet, yet still maintain its apparmor confinement, one would create/edit `/etc/firejail/curl.local` to include the lines.

```
# enable internet for curl
ignore net
```

Since `curl.local` is read before `globals.local`, `ignore net` overrides `net none`, and, as a bonus, the above changes would be persistent across future updates.

Testing profiles

Firejail's built in audit feature allows the user to find gaps in a security profile by replacing the program to be sandboxed with a test program. By default, firejail uses the `faudit` program distributed with Firejail. (Note: A custom test program supplied by the user can also be used.) Examples:

1. Run the default audit program: `$ firejail --audit transmission-gtk`
2. Run a custom audit program:
   `$ firejail --audit=~/sandbox-test transmission-gtk`

In the examples above, the sandbox configures the transmission-gtk profile and starts the test program. The real program, transmission-gtk, will not be started.

Note: The audit feature is not implemented for --x11 commands.

Firejail with Apparmor

Since 0.942, **firejail-apparmor (https://aur.archlinux.org/packages/firejail-apparmor/)**[AUR], has supported more direct integration with Apparmor through a generic apparmor profile. During installation, the profile, `firejail-default`, is placed in `/etc/apparmor.d` directory, and needs to be loaded into the kernel by running the following command as root:

```
# aa-enforce firejail-default
```

To quote the manual:

> *The installed profile tries to replicate some advanced security features inspired by kernel-based Grsecurity:*
>
> *- Prevent information leakage in /proc and /sys directories.The resulting filesystem is barely enough for running commands such as "top" and "ps aux".*
>
> *- Allow running programs only from well-known system paths, such as /bin, /sbin, /usr/bin etc. Running programs and scripts from user home or other directories writable by the user is not allowed.*
>
> *- Disable D-Bus. D-Bus has long been a huge security hole, and most programs don't use it anyway. You should have no problems running Chromium or Firefox.*

With the release of 0.9.50, local customisations of the apparmor profile are supported by editing the file `/etc/apparmor.d/local/firejail-local`

Apparmor usage

There are a number of ways to enable Apparmor confinement on top of a Firejail security profile:

- Pass the `--apparmor` flag to Firejail in the command line, eg `firejail --apparmor firefox`
- Use a custom profile.
- Enable Apparmor globally in `/etc/firejail/globals.local` and disable as needed through the use of `ignore apparmor` in `/etc/firejail/<ProgramName>.local`.

Firejail with Xephyr

Xephyr will allow you to sandbox **Xorg**. If you want to be able to resize windows, install a window manager such as **Openbox**.

`xephyr-screen WidthxHeight` can be set in `/etc/firejail/firejail.config` where `Width` and `Height` are in pixels and based on your screen resolution.

To open the sandbox:

```
$ firejail --x11 --net=device openbox
```

`device` is your active **network interface**. Then right click and select your applications to run.

Note: If you use **Unbound**, **dnsmasq**, **Pdnsd** or any other local cache as your resolver on 127.0.0.1 for example, you would leave `--net=device` out of the command as your network should work automatically.

A great guide can be found on the **Firejail Wordpress (https://firejail.wordpress.com/documentation-2/x11-guide/#configurexephyr)**.

According to the guide:

> The sandbox replaces the regular X11 server with Xpra or Xephyr server. This prevents X11 keyboard loggers and screenshot utilities from accessing the main X11 server.

Note that the statement:

> The only way to disable the abstract socket `@/tmp/.X11-unix/X0` is by using a network namespace. If for any reasons you cannot use a network namespace, the abstract socket will still be visible inside the sandbox. Hackers can attach keylogger and screenshot programs to this socket.

is incorrect, **xserverrc** can be edited to `-nolisten local` which disables the abstract sockets of X11 and helps isolate it.

Sandboxing a browser

Openbox can be configured to start a certain browser at startup. `program.profile` is the respective profile contained in `/etc/firejail`, and `--startup "command"` is the command line used to start the program. For example, to start Chromium in the sandbox:

```
$ firejail --x11 --profile=/etc/firejail/chromium.profile openbox --startup "chromium"
```

Tips and tricks

Paths containing spaces

If you need to reference, whitelist, or blacklist a directory within a custom profile, such as with **palemoon (https://aur.archlinux.org/packages/palemoon/)**[AUR], you must do so using the absolute path, without encapsulation or escapes:

```
/home/user/.moonchild productions
```

Private mode

Firejail also includes a one time private mode, in which no mounts are made in the chroots to your home directory. In doing this, you can execute applications without performing any changes to disk. For example, to execute okular in private mode, do the following:

```
$ firejail --seccomp --private okular
```

Troubleshooting

Some applications do not work properly with Firejail, and others simply require special configuration. In the instance any directories are disallowed or blacklisted for any given application, you may have to further edit the profile to enable nonstandard directories that said application needs to access. One example is wine; wine will not work with seccomp in most cases.

Other configurations exist; it is suggested you check out the man page for firejail to see them all, as firejail is in rapid development.

Remove Firejail symbolic links

To remove Firejail created symbolic links (e.g. reset to default):

```
# firecfg --clean
```

Verify if any leftovers of **Desktop entries** are still overruled by Firejail.

Desktop files

Some GUI application launchers (`.desktop` files) are coded using absolute paths to an executable, which circumvents firejail's symlink method of ensuring that it is being used. The *firecfg* tool includes an option to over-ride this on a per-user basis by copying the `.desktop` files from `/usr/share/applications/*.desktop` to `~/.local/share/applications/` and replacing the absolute paths with simple file names.

```
$ firecfg --fix
```

There may cases for which you need to manually modify the EXEC line of the `.desktop` file in `~/.local/share/applications/` to explicitly call Firejail.

Symbolic links

1. If used, any location database or hash table will need to be updated/reset.
2. Some applications, notably *Thunar*, run with only one instance. As a result, after symlinking firejail to the application, the profile may not be loaded until the next login.**[2] (https://github.com/netblue30/firejail/issues/1311)**

PulseAudio

Note: Using PulseAudio version 9.0 or later should fix this issue.

If Firejail causes **PulseAudio** issues with sandboxed applications **[3] (https://firejail.wordpress.com/support/known-problems/#pulseaudio)**, the following command may be used:

```
$ firecfg --fix-sound
```

This commands creates a custom `~/.config/pulse/client.conf` file for the *current* user with `enable-shm = no` and possible other workarounds.

Hidepid

If you have **hidepid** installed, Firemon can only be run as root. This, among other things, will cause problems with the Firetools GUI incorrectly reporting "Capabilities", "Protocols" and the status of "Seccomp". See **[4] (https://github.com/netblue30/firejail/issues/1564)**

Proprietary Nvidia drivers

Some users report problems when using Firejail and proprietary graphic drivers from **NVIDIA** (e.g. **[5] (https://github.com/netblue30/firejail/issues/1753)**, **[6] (https://github.com/netblue30/firejail/issues/879)** or **[7] (https://github.com/netblue30/firejail/issues/841)**). This can often be solved by disabling the `noroot` Firejail option in the application's profile file.

See also

- **Firejail GitHub project page (https://github.com/netblue30/firejail)**
- **bubblewrap**, a minimal alternative to Firejail

Retrieved from "https://wiki.archlinux.org/index.php?title=Firejail&oldid=508908"