containers / **bubblewrap**

Code | Issues 126 | Pull requests 29 | Actions | Projects | Security 1 | Insights

Low-level unprivileged sandboxing tool used by Flatpak and similar projects

Unknown, Unknown licenses found

Code of conduct

Security policy

4.1k stars · 246 forks · 54 watching · Branches · Activity · Custom properties · Tags

Public repository

14 Branches · 27 Tags · Go to file · Go to file · Add file · Code

smcv Prepare v0.11.0 · 9ca3b05 · 4 months ago

.github/workflows	workflows: Use latest version of actio...	5 months ago
ci	Remove autotools build system	6 months ago
completions	completions: Don't try to define more...	4 months ago
demos	Drop support for Python<3.4 in dem...	2 years ago
packaging	Remove trailing whitespace	4 years ago
tests	Add --overlay and related options	4 months ago
.dir-locals.el	Add .editorconfig and .dir-locals.el	9 years ago
.editorconfig	Remove trailing whitespace	4 years ago
CODE-OF-CONDUCT.md	Use HEAD to refer to other projects' d...	3 years ago
COPYING	Add LGPLv2+ COPYING (and LICENSE...	9 years ago
LICENSE	Add LGPLv2+ COPYING (and LICENSE...	9 years ago
NEWS.md	Prepare v0.11.0	4 months ago
README.md	Merge pull request #566 from TotalC...	5 months ago
SECURITY.md	README, SECURITY: Clarify that bubb...	2 years ago
bind-mount.c	Use stdbool.h for booleans	4 months ago
bind-mount.h	bind-mount: Include failing path in er...	3 years ago
bubblewrap.c	Add --overlay and related options	4 months ago
bubblewrap.jpg	Revert "README.md: Delete cat logo ...	7 years ago
bwrap.xml	Add --overlay and related options	4 months ago
	Prepare v0.11.0	4 months ago

meson.build		
meson_options.txt	meson: use boolean value for boolea...	7 months ago
network.c	Handle EINTR when doing I/O on files...	5 months ago
network.h	Add SPDX-License-Identifier for files t...	4 years ago
release-checklist.md	NEWS.md: Add items so far for 0.11.0	4 months ago
uncrustify.cfg	Add uncruftify config	9 years ago
uncrustify.sh	Add uncruftify config	9 years ago
utils.c	utils: Ensure that the buffer for struct ...	4 months ago
utils.h	Add --overlay and related options	4 months ago

Bubblewrap

Many container runtime tools like `systemd-nspawn` , `docker` , etc. focus on providing infrastructure for system administrators and orchestration tools (e.g. Kubernetes) to run containers.

These tools are not suitable to give to unprivileged users, because it is trivial to turn such access into a fully privileged root shell on the host.

User namespaces

There is an effort in the Linux kernel called user namespaces which attempts to allow unprivileged users to use container features. While significant progress has been made, there are still concerns about it, and it is not available to unprivileged users in several production distributions such as CentOS/Red Hat Enterprise Linux 7, Debian Jessie, etc.

See for example CVE-2016-3135 which is a local root vulnerability introduced by userns. This March 2016 post has some more discussion.

Bubblewrap could be viewed as setuid implementation of a *subset* of user namespaces. Emphasis on subset - specifically relevant to the above CVE, bubblewrap does not allow control over iptables.

The original bubblewrap code existed before user namespaces - it inherits code from xdg-app helper which in turn distantly derives from linux-user-chroot.

System security

The maintainers of this tool believe that it does not, even when used in combination with typical software installed on that distribution, allow privilege escalation. It may increase the ability of a logged in user to perform denial of service attacks, however.

In particular, bubblewrap uses `PR_SET_NO_NEW_PRIVS` to turn off setuid binaries, which is the traditional way to get out of things like chroots.

Sandbox security

bubblewrap is a tool for constructing sandbox environments. bubblewrap is not a complete, ready-made sandbox with a specific security policy.

Some of bubblewrap's use-cases want a security boundary between the sandbox and the real system; other use-cases want the ability to change the layout of the filesystem for processes inside the sandbox, but do not aim to be a security boundary. As a result, the level of protection between the sandboxed processes and the host system is entirely determined by the arguments passed to bubblewrap.

Whatever program constructs the command-line arguments for bubblewrap (often a larger framework like Flatpak, libgnome-desktop, sandwine or an ad-hoc script) is responsible for defining its own security model, and choosing appropriate bubblewrap command-line arguments to implement that security model.

Some aspects of sandbox security that require particular care are described in the Limitations section below.

Users

This program can be shared by all container tools which perform non-root operation, such as:

- Flatpak
- rpm-ostree unprivileged
- bwrap-oci

We would also like to see this be available in Kubernetes/OpenShift clusters. Having the ability for unprivileged users to use container features would make it significantly easier to do interactive debugging scenarios and the like.

Installation

bubblewrap is available in the package repositories of the most Linux distributions and can be installed from there.

If you need to build bubblewrap from source, you can do this with meson:

```
meson _builddir
meson compile -C _builddir
meson test -C _builddir
meson install -C _builddir
```

Usage

bubblewrap works by creating a new, completely empty, mount namespace where the root is on a tmpfs that is invisible from the host, and will be automatically cleaned up when the last process exits. You can then use commandline options to construct the root filesystem and process environment and command to run in the namespace.

There's a larger demo script in the source code, but here's a trimmed down version which runs a new shell reusing the host's `/usr`.

```
bwrap \
    --ro-bind /usr /usr \
    --symlink usr/lib64 /lib64 \
    --proc /proc \
    --dev /dev \
    --unshare-pid \
    --new-session \
    bash
```

This is an incomplete example, but useful for purposes of illustration. More often, rather than creating a container using the host's filesystem tree, you want to target a chroot. There, rather than creating the symlink `lib64 -> usr/lib64` in the tmpfs, you might have already created it in the target rootfs.

Sandboxing

The goal of bubblewrap is to run an application in a sandbox, where it has restricted access to parts of the operating system or user data such as the home directory.

bubblewrap always creates a new mount namespace, and the user can specify exactly what parts of the filesystem should be visible in the sandbox. Any such directories you specify mounted `nodev` by default, and can be made readonly.

Additionally you can use these kernel features:

User namespaces (CLONE_NEWUSER): This hides all but the current uid and gid from the sandbox. You can also change what the value of uid/gid should be in the sandbox.

IPC namespaces (CLONE_NEWIPC): The sandbox will get its own copy of all the different forms of IPCs, like SysV shared memory and semaphores.

PID namespaces (CLONE_NEWPID): The sandbox will not see any processes outside the sandbox. Additionally, bubblewrap will run a trivial pid1 inside your container to handle the requirements of reaping children in the sandbox. This avoids what is known now as the Docker pid 1 problem.

Network namespaces (CLONE_NEWNET): The sandbox will not see the network. Instead it will have its own network namespace with only a loopback device.

UTS namespace (CLONE_NEWUTS): The sandbox will have its own hostname.

Seccomp filters: You can pass in seccomp filters that limit which syscalls can be done in the sandbox. For more information, see Seccomp.

Limitations

As noted in the Sandbox security section above, the level of protection between the sandboxed processes and the host system is entirely determined by the arguments passed to bubblewrap. Some aspects that require special care are noted here.

- If you are not filtering out `TIOCSTI` commands using seccomp filters, argument `--new-session` is needed to protect against out-of-sandbox command execution (see CVE-2017-5226).
- Everything mounted into the sandbox can potentially be used to escalate privileges. For example, if you bind a D-Bus socket into the sandbox, it can be used to execute commands via systemd. You can use xdg-dbus-proxy to filter D-Bus communication.
- Some applications deploy their own sandboxing mechanisms, and these can be restricted by the constraints imposed by bubblewrap's sandboxing. For example, some web browsers which configure their child proccesses via seccomp to not have access to the filesystem. If you limit the syscalls and don't allow the seccomp syscall, a browser cannot apply these restrictions. Similarly, if these rules were compiled into a file that is not available in the sandbox, the browser cannot load these rules from this file and cannot apply these restrictions.

Related project comparison: Firejail

Firejail is similar to Flatpak before bubblewrap was split out in that it combines a setuid tool with a lot of

README Code of conduct License License Security

The bubblewrap authors believe it's much easier to audit a small setuid program, and keep features such as Pulseaudio filtering as an unprivileged process, as now occurs in Flatpak.

Also, @cgwalters thinks trying to whitelist file paths is a bad idea given the myriad ways users have to manipulate paths, and the myriad ways in which system administrators may configure a system. The bubblewrap approach is to only retain a few specific Linux capabilities such as `CAP_SYS_ADMIN`, but to always access the filesystem as the invoking uid. This entirely closes TOCTTOU attacks and such.

Related project comparison: Sandstorm.io

Sandstorm.io requires unprivileged user namespaces to set up its sandbox, though it could easily be adapted to operate in a setuid mode as well. @cgwalters believes their code is fairly good, but it could still make sense to unify on bubblewrap. However, @kentonv (of Sandstorm) feels that while this makes sense in principle, the switching cost outweighs the practical benefits for now. This decision could be re-evaluated in the future, but it is not being actively pursued today.

Related project comparison: runc/binctr

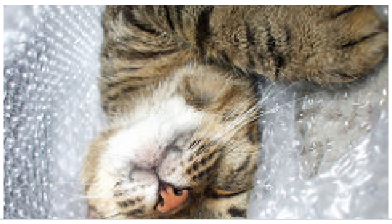
runC is currently working on supporting rootless containers, without needing `setuid` or any other privileges during installation of runC (using unprivileged user namespaces rather than `setuid`), creation, and management of containers. However, the standard mode of using runC is similar to systemd nspawn in that it is tooling intended to be invoked by root.

The bubblewrap authors believe that runc and systemd-nspawn are not designed to be made setuid, and are distant from supporting such a mode. However with rootless containers, runC will be able to fulfill certain usecases that bubblewrap supports (with the added benefit of being a standardised and complete OCI runtime).

binctr is just a wrapper for runC, so inherits all of its design tradeoffs.

What's with the name?!

The name bubblewrap was chosen to convey that this tool runs as the parent of the application (so wraps it in some sense) and creates a protective layer (the sandbox) around it.



Releases 23

0.11.0 Latest
on Oct 30, 2024

+ 22 releases

Packages

No packages published

Contributors 61

+ 47 contributors

Languages

● C 71.7% ● Shell 16.6% ● Python 8.1% ● Meson 3.6%