






LXC

LXC is the well-known and heavily tested low-level Linux container runtime. It is in active development since 2008 and has proven itself in critical production environments world-wide. Some of its core contributors are the same people that helped to implement various well-known containerization features inside the Linux kernel.

Status

Type	Service	Status
CI (Linux)	GitHub	 Fuzzing with OSS-fuzz passing
CI (Linux)	Jenkins	 build passing
Project status	CII Best Practices	 Fuzzing with OSS-fuzz passing
Fuzzing	OSS-Fuzz	 oss-fuzz fuzzing
Fuzzing	CIFuzz	 Fuzzing with OSS-fuzz passing

System Containers

LXC's main focus is system containers. That is, containers which offer an environment as close as possible as the one you'd get from a VM but without the overhead that comes with running a separate kernel and simulating all the hardware.

This is achieved through a combination of kernel security features such as namespaces, mandatory access control and control groups.

Unprivileged Containers

Unprivileged containers are containers that are run without any privilege. This requires support for user namespaces in the kernel that the container is run on. LXC was the first runtime to support unprivileged containers after user namespaces were merged into the mainline kernel.

In essence, user namespaces isolate given sets of UIDs and GIDs. This is achieved by establishing a mapping between a range of UIDs and GIDs on the host to a different (unprivileged) range of UIDs and GIDs in the container. The kernel will translate this mapping in such a way that inside the container all UIDs and GIDs appear as you would expect from the host whereas on the host these UIDs and GIDs are in fact unprivileged. For example, a process running as UID and GID 0 inside the container might appear as UID and GID 100000 on the host. The implementation and working details can be gathered from the corresponding user namespace man page.

Since unprivileged containers are a security enhancement they naturally come with a few restrictions enforced by the kernel. In order to provide a fully functional unprivileged container LXC interacts with 3 pieces of setuid code:

- `lxc-user-nic` (setuid helper to create a veth pair and bridge it on the host)
- `newuidmap` (from the shadow package, sets up a uid map)
- `newgidmap` (from the shadow package, sets up a gid map)

Everything else is run as your own user or as a uid which your user owns.

In general, LXC's goal is to make use of every security feature available in the kernel. This means LXC's configuration management will allow experienced users to intricately tune LXC to their needs.

A more detailed introduction into LXC security can be found under the following link

- <https://linuxcontainers.org/lxc/security/>

Removing all Privilege

In principle LXC can be run without any of these tools provided the correct configuration is applied. However, the usefulness of such containers is usually quite restricted. Just to highlight the two most common problems:

1. Network: Without relying on a `setuid` helper to setup appropriate network devices for an unprivileged user (see LXC's `lxc-user-nic` binary) the only option is to share the network namespace with the host. Although this should be secure in principle, sharing the host's network namespace is still one step of isolation less and increases the attack vector. Furthermore, when host and container share the same network namespace the kernel will refuse any `sysfs` mounts. This usually means that the `init` binary inside of the container will not be able to boot up correctly.
2. User Namespaces: As outlined above, user namespaces are a big security enhancement. However, without relying on privileged helpers users who are unprivileged on the host are only permitted to map their own UID into a container. A standard POSIX system however, requires 65536 UIDs and GIDs to be available to guarantee full functionality.

Configuration

LXC is configured via a simple set of keys. For example,

- `lxc.rootfs.path`
- `lxc.mount.entry`

LXC namespaces configuration keys by using single dots. This means complex configuration keys such as `lxc.net.0` expose various subkeys such as `lxc.net.0.type`, `lxc.net.0.link`, `lxc.net.0.ipv6.address`, and others for even more fine-grained configuration.

LXC is used as the default runtime for Incus, a container hypervisor exposing a well-designed and stable REST-api on top of it.

Kernel Requirements

LXC runs on any kernel from 2.6.32 onwards. All it requires is a functional C compiler. LXC works on all architectures that provide the necessary kernel features. This includes (but isn't limited to):

- i686
- x86_64
- ppc, ppc64, ppc64le

-
- riscv64
 - s390x
 - armv7l, arm64
 - loongarch64

LXC also supports at least the following C standard libraries:

- glibc
- musl
- bionic (Android's libc)

Backwards Compatibility

LXC has always focused on strong backwards compatibility. In fact, the API hasn't been broken from release 1.0.0 onwards. Main LXC is currently at version 4.*.*.

Reporting Security Issues

The LXC project has a good reputation in handling security issues quickly and efficiently. If you think you've found a potential security issue, please report it by e-mail to all of the following persons:

- serge (at) hallyn (dot) com
- stgraber (at) ubuntu (dot) com
- brauner (at) kernel (dot) org

For further details please have a look at

- <https://linuxcontainers.org/lxc/security/>

Becoming Active in LXC development

We always welcome new contributors and are happy to provide guidance when necessary. LXC follows the kernel coding conventions. This means we only require that each commit includes a [Signed-off-by](#) line. The coding style we use is identical to the one used by the Linux kernel. You can find a detailed introduction at:

- <https://www.kernel.org/doc/html/v4.10/process/coding-style.html>

and should also take a look at the CONTRIBUTING file in this repo.

If you want to become more active it is usually also a good idea to show up in the LXC IRC channel #lxc-dev on irc.libera.chat. We try to do all development out in the open and discussion of new features or bugs is done either in appropriate GitHub issues or on IRC.

When thinking about making security critical contributions or substantial changes it is usually a good idea to ping the developers first and ask whether a PR would be accepted.

Semantic Versioning

LXC and its related projects strictly adhere to a semantic versioning scheme.

Downloading the current source code

Source for the latest released version can always be downloaded from

- <https://linuxcontainers.org/lxc/downloads/>

You can browse the up to the minute source code and change history online

- <https://github.com/lxc/lxc>

Building LXC

Without considering distribution specific details a simple

```
1 meson setup -Dprefix=/usr build
2 meson compile -C build
```

is usually sufficient.

Getting help

When you find you need help, the LXC projects provides you with several options.

Discuss Forum

We maintain a discuss forum at

- <https://discuss.linuxcontainers.org/>

where you can get support.

IRC

You can find us in #lxc on irc.libera.chat.

Mailing Lists

You can check out one of the two LXC mailing list archives and register if interested:

- <http://lists.linuxcontainers.org/listinfo/lxc-devel>
- <http://lists.linuxcontainers.org/listinfo/lxc-users>