How to harden security for LXD

To increase the security posture of your LXD deployment, review the following hardening recommendations and apply those relevant to your setup.

General

Use a supported version

Use only supported LTS releases or the latest feature release of LXD, and ensure that you update it regularly to receive security updates and bugfixes. See: Releases.

Delete unused resources

Delete unused networks and storage pools to reduce the attack surface.

Access

Secure the `lxd` group

Users in the `lxd` group who access LXD through the local Unix socket are given full administrative control over LXD. Thus, ensure that only trusted users are members of the `lxd` group (or any custom group you configure via `snap.lxd.daemon.group`). Audit group membership regularly.

Also see: Use a restricted group for non-admin users.

Harden remote API access

For Remote API authentication, LXD can use either TLS client certificates or OpenID Connect:

- Client certificates:
  - Ensure that only clients with certificates issued by your trusted Certificate Authority (CA) can connect. The `core.trust_ca_certificates` option is `false` by default. To prevent auto-trusting of CA-signed certificates, ensure it remains disabled.
  - Regularly audit and remove unused client certificates from the trust store.
  - Ensure that private CAs issue short-lived certificates.
  - When using a PKI system, regularly audit and revoke unused client certificates using a certificate revocation list.
- OpenID Connect:
  - Only set `oidc.client.secret` if required by the identity provider.
  - Configure your OIDC provider to issue short-lived access tokens.
  - Require multi-factor authentication (MFA) in your identity provider.

For Remote API authorization, use Restricted TLS certificates or Fine-grained authorization where relevant to your setup.

Refer to the Remote API authentication and Remote API authorization pages for details.

Decrease token expiry

Decrease the expiry times for LXD cluster join tokens and remote authentication tokens, such as to 15 minutes each:

```
sudo lxc config set cluster.join_token_expiry 15M
sudo lxc config set core.remote_token_expiry 15M
```

Network security

Control traffic on LXD networks.

Configure ACLs

Network Access Control Lists (ACLs) are used to control traffic between instances and external networks, as well as traffic between instances on the same network. Set ACL rules to limit traffic to only what is necessary.

Limit network exposure

By default, LXD is only accessible locally through a Unix socket. If you need to expose LXD to the network, you must set the LXD server's `core.https_address`. To reduce the attack surface, do not set this address to a port alone.

Instead, use a trusted IP address on the LXD management interface along with a port, such as `192.0.2.10:8443`. If you only need local HTTPS access, use the loopback address and port, such as `127.0.0.1:8443`.

Instance security

Along with the recommendations below, review all instance security options for further options that might be relevant to your setup.

Rather than applying these options on a per-instance basis, use either Projects, profiles, or both. See the section on using profiles below.

Use unprivileged containers

By default, LXD containers are unprivileged. If you need to use privileged containers, make sure to put appropriate security measures in place. For more information, see: Container security.

Set instance resource limits

There are multiple Resource limits that can be configured for instances. To decrease the potential damage from DoS attacks, set reasonable limits.

This is especially important for containers and their `limits.cpu`, `limits.memory`, and `limits.processes` options, which by default are set without limits. Review the Resource limits reference guide for other options you might want to restrict.

Disable container nesting

The instance configuration option `security.nesting` enables nested container capability. This increases complexity and can broaden the attack surface. The default for this setting is `false`. Do not set this to `true` unless absolutely needed.

Setting this option to `true` is especially dangerous in combination with `security.privileged` set to `true` because it provides root access to the host.

Isolate containers

If a set of containers do not need to share data with each other, enable the instance option `security.idmap.isolated` on each one. This configures them to use unique UID/GID maps, preventing potential DoS attacks from one container to another. Only unprivileged containers can use this option.

Use profiles

Instead of applying Instance options on a per-instance basis, use either Projects, profiles, or both. This enables you to use a consistent hardened configuration.

The set of commands to create and use a profile below are provided as an example only, including the instance options explicitly mentioned in this guide. Review all instance options and decide if there are other options you want to set for your hardened profile.

```
sudo lxc profile create hardened1
sudo lxc profile set hardened1 limits.cpu=2 limits.memory=4GiB limits.proce
sudo lxc profile set hardened1 security.idmap.isolated=true security.nestin
sudo lxc profile add <my-container> hardened1
```

Device security

Limit device passthrough

PCI, USB, and disk device passthroughs give the container significant access to the host. Avoid adding devices to instances unless strictly necessary. Set disk device mounts to `readonly` where possible.

Prevent spoofing

With bridged NICs, the default configuration allows MAC or IP spoofing. For details on how to prevent this, see Bridged NIC security.

Storage device security

The Linux kernel might ignore mount options if a block-based filesystem (like `ext4`) is already mounted with different options. Thus, sharing the same disk device across multiple storage pools can lead to unexpected mount behavior.

To avoid security issues, either dedicate a disk device per storage pool or ensure that all pools sharing a device use the same mount options. For more information, see the Security considerations section of the Storage drivers reference guide.

Logging

Increase logging and regularly audit the logs for suspicious activity.

Use system logging

Enable system logging for the LXD daemon and set it to the `verbose` level:

```
sudo snap set lxd daemon.syslog=true
sudo snap set lxd daemon.verbose=true
```

Regularly check these logs using:

```
sudo snap logs lxd.daemon
```

By default, only the last 10 lines are output. To see more, use the `-n=[all|<#>]` flag.

For example, to see all logs, run:

```
sudo snap logs -n=all lxd.daemon
```

Use `auditd` rules

Use `auditd` rules to track LXD command execution and configuration file changes.

Configure the audit daemon to track all commands to the LXD daemon:

```
-a always,exit -F path=/snap/bin/lxc -p x -k lxd_execution
-a always,exit -F path=/snap/bin/lxd -p x -k lxd_execution
-a always,exit -F path=/snap/bin/lxd.buginfo -p x -k lxd_execution
-a always,exit -F path=/snap/bin/lxd.check-kernel -p x -k lxd_execution
-a always,exit -F path=/snap/bin/lxd.lxc -p x -k lxd_execution
```

`lxc monitor` and Loki

The `lxc monitor` command is used to view information about logging and life cycle LXD events. Consider using a dedicated system that allows you to keep a record of these events, such as Loki. See: How to send logs to Loki.

Multi-user environment

These settings are relevant if your LXD server is used by multiple users, such as in a lab setting.

Use a restricted group for non-admin users

By default, both the `daemon.group` and `daemon.user.group` are set to `lxd`. This gives all users in the `lxd` group full local access to LXD through the Unix socket. This includes the ability to attach file system paths or devices to any instance, or tweak any instance's security features.

Only users who are trusted with `sudo` access to your system should be in the `daemon.group`. Define and use a separate group for users who should not have admin access, such as `lxdusers`:

```
sudo groupadd lxdusers
sudo snap set lxd daemon.user.group=lxdusers
```

Confine users to projects

You can confine users to specific projects, which can be configured with stricter restrictions to prevent misuse. For details, see: Confine users to specific LXD projects via Unix socket, Instances grouping with projects, and Restricted TLS certificates.

Prevent name leakage

The default server configuration makes it possible to list all cgroups on a system, and by extension, all running containers. Prevent container name leakage by blocking access to `/sys/kernel/slab` and `/proc/sched_debug` before you start any containers. To do so, run:

```
chmod 400 /proc/sched_debug
chmod 700 /sys/kernel/slab/
```

Harden the LXD host OS

To harden your deployment, also harden the host's operating system (OS). These are some ways you can harden the host OS:

- Keep your OS updated and install all available security patches.
- Use a firewall to drop unexpected inbound traffic and restrict outbound traffic as needed. Ensure only the necessary ports are open.
- For Ubuntu systems, subscribe to Ubuntu Pro.
- Use the latest CIS hardening benchmarks for your OS.

Ubuntu CIS hardening

For Ubuntu LTS releases subscribed to Ubuntu Pro, use the Ubuntu Security Guide (USG) tool for CIS hardening. The tool can audit the host system and fix many issues automatically. Depending on how your system is configured, there might be other issues that you must remediate manually.

There are known issues with three of the auditing tool's rule IDs when auditing LXD hosts with the `cis_level1_server` profile. One is that it generates a false failure report for the following rule ID, flagging that no UEFI boot loader password is set even when it is:

```
xccdf_org.ssgproject.content_rule_grub2_uefi_password
```

As long as you have set this password and can confirm that the UEFI boot process requests it, you can ignore this failure report.

Furthermore, if the Ubuntu system is running LXD containers, the USG audit will report failure on the following rule IDs:

```
xccdf_org.ssgproject.content_rule_no_files_unowned_by_user
xccdf_org.ssgproject.content_rule_file_permissions_ungroupowned
```

By design, LXD's unprivileged containers run inside a user namespace for greater isolation. This causes some files and directories under `/sys/fs/cgroup/lxc.payload.<container_name>` to appear as having no owner. Since this is expected, the USG tool's failure report for this can be ignored.

You can customize the tool's CIS profile to always ignore these three rule IDs. To do so, follow the instructions in the Customizing CIS profiles section of the Ubuntu security documentation.

Related topics

How-to guides:

- How to configure your firewall
- How to confine users to specific projects

Explanation:

- Security
- Remote API authentication
- Remote API authorization

Reference:

- Instance-level security options