CEPHX GUIDE

Ceph provides two authentication modes:

- **None:** Any user can access data without authentication.
- **Cephx**: Ceph requires user authentication in a manner similar to Kerberos.

If you disable cephx, you do not need to generate keys using the procedures described here. If you re-enable cephx and have already generated keys, you do not need to generate the keys again.

Important: The cephx protocol does not address data encryption in transport (e.g., SSL/TLS) or encryption at rest.

For additional information, see our Cephx Intro, our Cephx Configuration Reference and ceph-authtool manpage.

CONFIGURING CEPHX

There are several important procedures you must follow to enable the cephx protocol for your Ceph cluster and its daemons:

1. You must generate a secret key for the default `client.admin` user so the administrator can execute Ceph commands.
2. You must generate a monitor secret key and distribute it to all monitors in the cluster.
3. You must follow the remaining steps in Enabling Cephx to enable authentication.

See the Cephx Configuration Reference for additional details.

THE `CLIENT.ADMIN` KEY

When you first install Ceph, each Ceph command you execute on the command line assumes that you are the `client.admin` default user. When running Ceph with cephx enabled, you need to have a key for the `client.admin` user to run ceph commands as the administrator.

Important: To run Ceph commands on the command line with cephx enabled, you need to create a key for the `client.admin` user, and create a secret file under `/etc/ceph`.

The following command will generate and register a `client.admin` key on the monitor with admin capabilities and write it to a keyring on the local file system. If the key already exists, its current value will be returned.

```
sudo ceph auth get-or-create client.admin mds 'allow' osd 'allow *' mon 'allow *' > /etc/ceph
```

See Enabling Cephx step 1 for stepwise details to enable cephx.

MONITOR KEYRINGS

Ceph requires a keyring for the monitors. Use the ceph-authtool command to generate a secret monitor key and keyring.

```
sudo ceph-authtool {keyring} --create-keyring --gen-key -n mon.
```

A cluster with multiple monitors must have identical keyrings for all monitors. So you must copy the keyring to each monitor host under the following directory:

```
/var/lib/ceph/mon/$cluster-$id
```

See Enabling Cephx step 2 and 3 for stepwise details to enable cephx.

ENABLING CEPHX

When cephx is enabled, Ceph will look for the keyring in the default search path, which includes `/etc/ceph/keyring`. You can

override this location by adding a `keyring` option in the `[global]` section of your Ceph configuration file, but this is not recommended.

Execute the following procedures to enable cephx on a cluster with cephx disabled. If you (or your deployment utility) have already generated the keys, you may skip the steps related to generating keys.

1. Create a `client.admin` key, and save a copy of the key for your client host:

```
ceph auth get-or-create client.admin mon 'allow *' mds 'allow *' osd 'allow *' -o /etc/ce

**Warning:** This will clobber any existing ``/etc/ceph/keyring`` file. Be careful!
```

2. Generate a secret monitor `mon.` key:

```
ceph-authtool --create --gen-key -n mon. /tmp/monitor-key
```

3. Copy the mon keyring into a `keyring` file in every monitor's `mon data` directory:

```
cp /tmp/monitor-key /var/lib/ceph/mon/ceph-a/keyring
```

4. Generate a secret key for every OSD, where `{$id}` is the OSD number:

```
ceph auth get-or-create osd.{$id} mon 'allow rwx' osd 'allow *' -o /var/lib/ceph/osd/ceph
```

5. Generate a secret key for every MDS, where `{$id}` is the MDS letter:

```
ceph auth get-or-create mds.{$id} mon 'allow rwx' osd 'allow *' mds 'allow *' -o /var/lib
```

6. Enable cephx authentication for versions `0.51` and above by setting the following options in the `[global]` section of your Ceph configuration file:

```
auth cluster required = cephx
auth service required = cephx
auth client required = cephx
```

7. Or, enable cephx authentication for versions `0.50` and below by setting the following option in the `[global]` section of your Ceph configuration file:

```
auth supported = cephx
```

Deprecated since version 0.51.

1. Start or restart the Ceph cluster.

```
sudo service ceph -a start
sudo service ceph -a restart
```

DISABLING CEPHX

The following procedure describes how to disable Cephx. If your cluster environment is relatively safe, you can offset the computation expense of running authentication. **We do not recommend it.** However, it may be easier during setup and/or troubleshooting to temporarily disable authentication.

1. Disable cephx authentication for versions `0.51` and above by setting the following options in the `[global]` section of your Ceph configuration file:

```
auth cluster required = none
auth service required = none
auth client required = none
```

2. Or, disable cephx authentication for versions `0.50` and below (deprecated as of version 0.51) by setting the following option in the `[global]` section of your Ceph configuration file:

```
auth supported = none
```

3. Start or restart the Ceph cluster.

```
sudo service ceph -a start
sudo service ceph -a restart
```

DAEMON KEYRINGS

With the exception of the monitors, Ceph generates daemon keyrings in the same way that it generates user keyrings. By default, the daemons store their keyrings inside their data directory. The default keyring locations, and the capabilities necessary for the daemon to function, are shown below.

`ceph-mon`

Location: `$mon_data/keyring`
Capabilities: N/A

`ceph-osd`

Location: `$osd_data/keyring`
Capabilities: `mon 'allow rwx' osd 'allow *'`

`ceph-mds`

Location: `$mds_data/keyring`
Capabilities: `mds 'allow rwx' mds 'allow *' osd 'allow *'`

`radosgw`

Location: `$rgw_data/keyring`
Capabilities: `mon 'allow r' osd 'allow rwx'`

Note that the monitor keyring contains a key but no capabilities, and is not part of the cluster `auth` database.

The daemon data directory locations default to directories of the form:

```
/var/lib/ceph/$type/$cluster-$id
```

For example, `osd.12` would be:

```
/var/lib/ceph/osd/ceph-12
```

You can override these locations, but it is not recommended.

CEPHX ADMINISTRATION

Cephx uses shared secret keys for authentication, meaning both the client and the monitor cluster have a copy of the client's secret key. The authentication protocol is such that both parties are able to prove to each other they have a copy of the key without actually revealing it. This provides mutual authentication, which means the cluster is sure the user possesses the secret key, and the user is sure that the cluster has a copy of the secret key.

Default users and pools are suitable for initial testing purposes. For test bed and production environments, you should create users and assign pool access to the users.

ADD A KEY

Keys enable a specific user to access the monitor, metadata server and cluster according to capabilities assigned to the key. Capabilities are simple strings specifying some access permissions for a given server type. Each server type has its own string. All capabilities are simply listed in {type} and {capability} pairs on the command line:

```
sudo ceph auth get-or-create client.{username} {daemon1} {cap1} {daemon2} {cap2} ...
```

For example, to create a user `client.foo` with access 'rw' for daemon type 'osd' and 'r' for daemon type 'mon':

```
sudo ceph auth get-or-create client.foo osd 'allow rw' mon 'allow r' > keyring.foo
```

Note: User names are associated to user types, which include `client` `osd`, `mon`, and `mds`. In most cases, you will be creating keys for `client` users.

After you add a key to the cluster keyring, go to the relevant client(s) and copy the keyring from the cluster host to the client(s).

```
sudo scp {user}@{ceph-cluster-host}:/etc/ceph/ceph.keyring /etc/ceph/ceph.keyring
```

Tip: Ensure the `ceph.keyring` file has appropriate permissions set (e.g., `chmod 644`) on your client machine.

DELETE A KEY

To delete a key for a user or a daemon, use `ceph auth del`:

```
ceph auth del {daemon-type}.{ID|username}
```

Where `{daemon-type}` is one of `client`, `osd`, `mon`, or `mds`, and `{ID|username}` is the ID of the daemon or the username.

After you delete a key from the cluster keyring, go to the relevant client(s) and copy the keyring from the cluster host to the client(s).

```
sudo scp {user}@{ceph-cluster-host}:/etc/ceph/ceph.keyring /etc/ceph/ceph.keyring
```

Tip: Ensure the `ceph.keyring` file has appropriate permissions set (e.g., `chmod 644`) on your client machine.

LIST KEYS IN YOUR CLUSTER

To list the keys registered in your cluster:

```
sudo ceph auth list
```

CEPHX COMMANDLINE OPTIONS

When Ceph runs with Cephx enabled, you must specify a user name and a secret key on the command line. Alternatively, you may use the `CEPH_ARGS` environment variable to avoid re-entry of the user name and secret.

```
ceph --id {user-name} --keyring=/path/to/secret [commands]
```

For example:

```
ceph --id client.admin --keyring=/etc/ceph/ceph.keyring [commands]
```

Ceph supports the following usage for user name and secret:

`--id` | `--user`

Description: Ceph identifies users with a type and an ID (e.g., `TYPE.ID` or `client.admin`, `client.user1`). The `id`, `name` and `-n` options enable you to specify the ID portion of the user name (e.g., `admin`, `user1`, `foo`, etc.). You can specify the user with the `--id` and omit the type. For example, to specify user `client.foo` enter the following:

```
ceph --id foo --keyring /path/to/keyring health
ceph --user foo --keyring /path/to/keyring health
```

`--name`

Description: Ceph identifies users with a type and an ID (e.g., `TYPE.ID` or `client.admin`, `client.user1`). The `--name` and `-n` options enables you to specify the fully qualified user name. You must specify the user type (typically `client`) with the user ID. For example:

```
ceph --name client.foo --keyring /path/to/keyring health
ceph -n client.foo --keyring /path/to/keyring health
```

`--keyring`

Description: The path to the keyring containing one or more user name and secret. The `--secret` option provides the same functionality, but it does not work with Ceph RADOS Gateway, which uses `--secret` for another purpose. You may retrieve a keyring with `ceph auth get-or-create` and store it locally. This is a preferred approach, because you can switch user names without switching the keyring path. For example:

```
sudo rbd map foo --pool rbd myimage --id client.foo --keyring /path/to/keyring
```

`--keyfile`

Description: The path to the key file containing the secret key for the user specified by `--id`, `--name`, `-n`, or `--user`. You may retrieve the key for a specific user with `ceph auth get` and store it locally. Then, specify the path to the keyfile. For example:

```
sudo rbd map foo --pool rbd myimage --id client.foo --keyfile /path/to/file
```

Note: Add the user and secret to the `CEPH_ARGS` environment variable so that you don't need to enter them each time. You can override the environment variable settings on the command line.

BACKWARD COMPATIBILITY

New in version Bobtail.

In Ceph Argonaut v0.48 and earlier versions, if you enable cephx authentication, Ceph only authenticates the initial communication between the client and daemon; Ceph does not authenticate the subsequent messages they send to each other, which has security implications. In Ceph Bobtail and subsequent versions, Ceph authenticates all ongoing messages between the entities using the session key set up for that initial authentication.

We identified a backward compatibility issue between Argonaut v0.48 (and prior versions) and Bobtail (and subsequent versions). During testing, if you attempted to use Argonaut (and earlier) daemons with Bobtail (and later) daemons, the Argonaut daemons did not know how to perform ongoing message authentication, while the Bobtail versions of the daemons insist on authenticating message traffic subsequent to the initial request/response–making it impossible for Argonaut (and prior) daemons to interoperate with Bobtail (and subsequent) daemons.

We have addressed this potential problem by providing a means for Argonaut (and prior) systems to interact with Bobtail (and subsequent) systems. Here's how it works: by default, the newer systems will not insist on seeing signatures from older systems that do not know how to perform them, but will simply accept such messages without authenticating them. This new default behavior provides the advantage of allowing two different releases to interact. **We do not recommend this as a long term solution**. Allowing newer daemons to forgo ongoing authentication has the unfortunate security effect that an attacker with control of some of your machines or some access to your network can disable session security simply by claiming to be unable to sign messages.

> **Note:** Even if you don't actually run any old versions of Ceph, the attacker may be able to force some messages to be accepted unsigned in the default scenario. While running Cephx with the default scenario, Ceph still authenticates the initial communication, but you lose desirable session security.

If you know that you are not running older versions of Ceph, or you are willing to accept that old servers and new servers will not be able to interoperate, you can eliminate this security risk. If you do so, any Ceph system that is new enough to support session authentication and that has Cephx enabled will reject unsigned messages. To preclude new servers from interacting with old servers, include the following in the `[global]` section of your Ceph configuration file directly below the line that specifies the use of Cephx for authentication:

```
cephx require signatures = true    ; everywhere possible
```

You can also selectively require signatures for cluster internal communications only, separate from client-facing service:

```
cephx cluster require signatures = true    ; for cluster-internal communication
cephx service require signatures = true    ; for client-facing service
```

An option to make a client require signatures from the cluster is not yet implemented.

We recommend migrating all daemons to the newer versions and enabling the foregoing flag at the nearest practical time so that you may avail yourself of the enhanced authentication.