

Ceph CVEs: auth service entities with insecure key types

Julien Laurenceau | Published 2026-08-24 | Updated 2026-10-09

<https://pepitedata.com/ceph-tentacle-squid-security-release/>

PepiteData | LinkedIn: Julien Laurenceau | X: @Julien_Lau



Release hygiene and key rotation are part of the Ceph cluster audits I run: [Ceph storage consulting](#).

[ERR] AUTH_INSECURE_SERVICE_KEY_TYPE: 46 auth service entities with insecure key types

Correction and warning, August 28, 2026: the Ceph documentation is wrong about automatic key rotation.

When I published this post I repeated the upstream documentation: `cephadm` automates daemon-key rotation on upgrade. That claim is false. I verified the facts below against a Ceph 20.2.4 cluster and the 20.2.4 source.

- **Key rotation is not automatic on a Ceph upgrade with `cephadm`.** The Tentacle documentation (and latest, as of this correction) presents orchestrator daemon-key rotation as available. It is not.
- **Ceph 20.2.4 source explicitly disables orchestrator daemon-key rotation.** On 20.2.4, `ceph orch daemon rotate-key` returns `Invalid command: rotate-key`, and the `cephadm` source raises

key rotation by orchestrator not supported in this release. Upstream disabled the command deliberately in [commit 00881b76c](#) because it could take daemons with client keys, such as NFS and NVMe-oF gateways, down.

- **No safe ceph auth rotate plus restart or redeploy equivalent exists.** The workaround this post originally recorded from the mailing list does not work. `ceph auth rotate` invalidates the old key immediately, so the daemon cannot reauthenticate until it is fully redeployed, and an OSD also carries its key in the bluestore label, which only `ceph-bluestore-tool set-label-key` updates in the documented procedure. I have removed that workaround from this post. The mon key is the documented exception, covered by the procedure below.
- **One layer does heal itself: the rotating service keys.** These are short lived session keys, not entity keys. Once the Monitors run 20.2.4 and `auth_service_cipher` is set to `aes256k`, new rotating keys use the new cipher and the legacy `aes` rotating keys expire on their own within a few hours on the default TTL, clearing `AUTH_INSECURE_ROTATING_SERVICE_KEY_TYPE` with no action. [Eugen Block spells this out on the mailing list](#).

Addendum, August 31, 2026: two families of keys, two different paths. A follow-up on the Ceph users list settled a distinction this post carried implicitly. Internal service keys and client keys are different families: the upgrade fixes the CVEs while `cephadm` leaves every client keyring untouched, so client rotation is a separate migration you schedule per client, not part of the upgrade window. See the new section in "What the Ceph users thread adds" below.

Addendum, October 9, 2026: the tested OSD rotation procedure is now in this post. The "no safe equivalent" statement above rules out one-command workarounds. The rotation I run on `cephadm` clusters, `rotate plus BlueStore label write` under host maintenance, is now spelled out in "Rotating OSD keys on a `cephadm` cluster" below.

Read the CVEs and the upgrade sequencing below as written. The mon key and the rotating service keys have a short documented procedure, and I spell it out below. The other entity keys and the client keys are the part with no safe one-command equivalent: budget for a manual, tested rotation, or wait for upstream to ship a working command.

On August 19, 2026, the Ceph project released two hotfix versions: Tentacle 20.2.4 and Squid 19.2.6. Both patch four CVEs in authentication and authorization code. Ceph asks all operators to upgrade as soon as possible.

That same month, French authorities disclosed several security incidents, including one at DGFIP involving data relating to 678,000 individuals and businesses. Some involved confirmed extraction. The scope of the France Titres and Education incidents was still under investigation when I wrote this. No source reviewed links Ceph or these CVEs to any of them. The comparison is operational context only: the release is a good moment to inspect identity, permissions, and detection on your own storage cluster.

Ceph security update at a glance

On August 19, 2026, Ceph released Tentacle 20.2.4 and Squid 19.2.6 to fix four authentication and authorization CVEs, and asks all operators to upgrade as soon as possible. Upgrading packages alone is not remediation: plan the CephX key migration to `aes256k` as well. `cephadm` does not rotate daemon keys on upgrade, client keyrings are never touched, and RGW multisite needs a coordinated sequence. Upstream sources reviewed do not report exploitation in the wild.

- **Affected releases:** Tentacle 20.2.4 and Squid 19.2.6 fix four authentication and authorization CVEs.

- **Immediate action:** Upgrade packages or container images, then plan the CephX key migration to aes256k. Package installation alone does not rotate every key, and nothing in 20.2.4 rotates them for you. See the correction above.
- **Operational dependencies:** cephadm does not rotate daemon keys on upgrade, despite what the documentation claims. Client keyrings are never touched: client rotation is a separate migration from the upgrade. Rook automates some client rotations with exclusions. RGW multisite requires a coordinated upgrade sequence.
- **Scope:** The upstream sources reviewed do not report exploitation in the wild, and no reviewed source connects these CVEs to the French public-sector incidents below.

What the four CVEs are

The four CVEs are CVE-2025-30156, a CephX authentication bypass; CVE-2026-39944, an RGW STS token escalation; CVE-2026-50152, a Monitor config-key store exposure; and CVE-2026-54330, an RGW SigV4 verification flaw. All four advisories come from Ceph's own documentation, and each is summarized below with its full advisory. Every flaw concerns authentication, signature verification, or authorization rather than data-path performance or replication correctness.

CVE-2025-30156: CephX authentication bypass. Legacy CephX aes keys use unauthenticated AES-128-CBC. The advisory says an attacker needs a compromised low-privilege key or a position from which to observe relevant CephX ciphertext. Under those conditions, the flaw can enable credential forgery and authentication bypass. The fix introduces a new aes256k key type based on AES256-CTS-HMAC-SHA384-192. Read the [Ceph advisory for CVE-2025-30156](#).

That prerequisite matters. Installing new packages without rotating legacy keys is not the whole remediation.

CVE-2026-39944: RGW STS token escalation. STS session tokens share the same AES-CBC root cause. Someone holding a valid unprivileged STS token can alter protected fields and escalate to RGW admin access. Read the [Ceph advisory for CVE-2026-39944](#).

CVE-2026-50152: Monitor config-key store exposure. A CephX identity with only `mon allow r` could read the entire Monitor config-key store, which may hold secrets. Read the [Ceph advisory for CVE-2026-50152](#).

CVE-2026-54330: RGW SigV4 verification flaw. A holder of a presigned PUT URL could add unsigned `x-amz-*` headers and gain capabilities beyond the signer's intent. Patched RGW rejects such headers when they are absent from the signed subset. Read the [Ceph advisory for CVE-2026-54330](#).

Two points stand out. First, all four flaws concern authentication, signature verification, or authorization. None concerns data-path performance or replication correctness. Second, the advisory for CVE-2025-30156 documents a researcher proof of concept. That is not evidence of exploitation in the wild, and the upstream sources reviewed do not report any.

The upgrade is not a one-line command

The upgrade is not a one-line command: package installation is only the first step, and CephX key migration is the rest. Read the upgrade guidance before scheduling anything. The announcement and documentation specify several steps: a key-upgrade and rotation procedure, a manual daemon-key rotation that cephadm does not automate, Rook exclusions to check, client and kernel upgrades for aes256k support, new auth health warnings during migration, and a coordinated sequence for RGW multisite.

- Review the new [CephX key-upgrade and rotation procedure](#) first.

- cephadm does not automate daemon-key rotation, despite the documentation. The `ceph orch daemon rotate-key` command is disabled in 20.2.4. Plan a manual rotation procedure and test it before the maintenance window.
- Rook automates some client-key rotation, with exclusions. Check them before you start.
- Client and kernel upgrades are recommended for aes256k support. Upstream Linux support began in kernel 7.0 and was backported to CentOS Stream 9 and 10. For other distributions, verify support with your vendor.
- Expect six new authentication-related health warnings and errors during migration. Ceph says this is normal, but operators must address them as keys are upgraded and rotated.
- Running RGW multisite? Set `rgw_sigv4_insecure=true` before upgrading. Restore it to false only after every cluster in the federation is patched.
- For CVE-2026-50152, cephadm users should rotate the cephadm SSH key. Guidance for rotating other config-key secrets is still forthcoming upstream.

Sources: the [Ceph release announcement](#) and the [cephadm SSH configuration guidance](#).

What the Ceph users thread adds

The Ceph users thread adds field experience and maintainer clarification to the upgrade documentation: the [Ceph users mailing-list archive for this urgent release](#) documents the operational questions that emerge after a test upgrade. It covers the split between internal service keys and client keys, shared client identity inventory, daemon-key rotation worklists, and the post-patch procedure for cephadm clusters. The maintained Ceph documentation remains the authority when it differs.

Two families of keys: internal service keys and client keys

The distinction that organizes the whole migration. CephX keys come in two families. Internal service keys belong to the cluster itself: the monitor, OSD, MDS and other daemon identities, plus the short lived rotating session keys the Monitors issue. Client keys are the keyrings you distribute outside the cluster: copies of `client.admin`, per node kernel client keyrings, and application identities such as the credentials libvirt or Proxmox consume. The upgrade path treats the two families differently, and conflating them is how a one hour patch grows into a multi day migration.

[Eugen Block drew the line in the follow-up thread](#), answering an operator who runs Proxmox nodes on Ceph storage and wants to defer key rotation: "If you upgrade to 20.2.4 the key rotation for the Ceph services will be done by the orchestrator automatically. But only for the Ceph services, no client keyrings will be touched by cephadm. So you can safely upgrade and fix the CVEs but can still use clients with the aes format keys."

Read the first half of that quote against the correction at the top of this post: on 20.2.4 the orchestrator does not rotate daemon entity keys, and the layer that renews on its own is the rotating session keys. The planning-relevant half is the second one. No client keyring is touched by cephadm, so clients keep their aes keys and keep authenticating after the upgrade, because aes stays in the allowed cipher list until you remove it. Client rotation is a separate migration you schedule per client, and [the documented procedure](#) covers the step Patrick Donnelly flags for it: after rotating a client key, "copy and import the key to each machine using that `client.$ID` credential." Your worklist follows the split: the internal keys size the cluster-side window below, the client keys size a sequence of per client cutovers that can start after the upgrade rather than during it.

Inventory shared client identities before the maintenance window

Patrick Donnelly clarified that one Ceph client identity cannot hold both aes and aes256k keys at the same time. That makes a shared `client.libvirt` or application key a migration constraint. For gradual replacement, create a distinct client identity and key for each node or workload, distribute it to the replacement node, test it, then retire the old shared identity only after its consumers have moved. Do the identity inventory before changing the default cipher.

Make a worklist for every daemon key that needs rotation

Nothing rotates these keys for you. The 20.2.4 source explicitly disables orchestrator daemon-key rotation: `ceph orch daemon rotate-key` returns `Invalid command: rotate-key`, and the `cephadm` code raises key rotation by orchestrator not supported in this release. Upstream disabled the command in [commit 00881b76c](#) because it could break daemons that carry client keys, such as NFS and NVMe-oF gateways.

The rotation workaround this post previously recorded from the thread, `ceph auth rotate <entity-name> --key-type aes256k` followed by `ceph orch daemon redeploy <daemon-name>`, is not a safe equivalent. Rotating the entity invalidates the old key at once, which leaves the daemon unable to reauthenticate until the redeploy completes. An OSD additionally carries its key in the bluestore label, and the documented procedure updates it with `ceph-bluestore-tool set-label-key`. Treat rotation as a designed migration: build a worklist of every CephX identity, daemon and client, and test the full sequence on a non-production cluster before the maintenance window. NFS exports can have separate keyrings and may need a rotation and reapply cycle, so test their availability impact first.

The post-patch procedure for cephadm clusters

What actually remains after patching is short. Once every daemon runs 20.2.4, the mon key, the cipher policy, and the rotating service keys all come down to three protective flags, seven commands, and a cleanup. Expect every OSD in the cluster to restart in a wave while the Monitors take the new key and the auth epoch advances, which is why the flags come first. Pick a low activity window: the flags prevent data movement, but every client crosses the auth epoch change at the same moment, and busy clients feel it as a latency spike and a burst of reconnections. I assembled this sequence for `cephadm` from the documented package-based procedure and the thread, with one correction the documentation buries: a monmap that predates 20.2.4 decodes with `auth_service_cipher` set to `aes`, so the service cipher must be set before wiping the rotating keys, otherwise the Monitors just regenerate aes rotating keys.

```
ceph osd set noout
ceph osd set norebalance
ceph health mute OSDMAP_FLAGS 2h

ceph orch upgrade check quay/ceph/ceph:v20.2.4
ceph config set mgr mgr/cephadm/container_image_base quay/ceph/ceph:v20.2.4
ceph orch upgrade start quay/ceph/ceph:v20.2.4

ceph mon set auth_allowed_ciphers aes,aes256k
ceph mon set auth_preferred_cipher aes256k
ceph auth rotate --key-type=aes256k mon. | tee mon.keyring
ceph orch restart mon
ceph mon set auth_service_cipher aes256k
ceph config set mon 'mon auth allow insecure key' false
ceph auth wipe-rotating-service-keys
```

Why each command, in order:

- `ceph osd set noout` covers the restart wave. OSDs that drop their mon session and come back are not marked out, so no recovery or data movement starts while the wave passes.
- `ceph osd set norebalance` freezes PG rebalancing for the duration. With `noout` it is belt and suspenders: together they guarantee the wave moves no data.

- `ceph health mute OSDMAP_FLAGS 2h` keeps dashboards and alerts quiet while the two flags are set. The warning comes back by itself after two hours if the flags are still set, which acts as a reminder rather than a nuisance.
- `ceph mon set auth_allowed_ciphers aes,aes256k` keeps both ciphers valid during the transition. On an upgraded cluster the Monitors usually already allow both, so this mainly makes the state explicit.
- `ceph mon set auth_preferred_cipher aes256k` makes aes256k the default for every key created from now on. A pre 20.2.4 monmap still carries aes here.
- `ceph auth rotate --key-type=aes256k mon.` rotates the single `mon.` key that all Monitors share, and tee saves the new keyring. This is the one `ceph auth rotate` that is safe to run directly: the Monitors' keyring files stay on disk as a fallback, and the authoritative copy moves into the Monitor auth database.
- `ceph orch restart mon` restarts each Monitor so it picks up the new key from the auth database. You do not need to touch keyring files, with one exception: a Monitor that was out of quorum during the rotation will not have the new key, and you must install the saved `mon.keyring` into its keyring file.
- `ceph mon set auth_service_cipher aes256k` sets the cipher for the rotating session keys the Monitors issue. This line has a gate: run it only once `AUTH_INSECURE_SERVICE_KEY_TYPE` is gone from `ceph health detail`, meaning every core daemon entity key is already aes256k. On the cluster from the thread, the upgrade path had rotated the OSD keys already. On a cluster where the cephadm upgrade finished cleanly, this setting was applied during the upgrade, so the command is confirmation as much as change.
- `ceph config set mon 'mon auth allow insecure key' false` prevents the creation of new aes keys. This matters because keeping aes in `auth_allowed_ciphers` flips the runtime default of `mon_auth_allow_insecure_key` to true, and with it comes the `AUTH_INSECURE_KEYS_CREATABLE` warning.
- `ceph auth wipe-rotating-service-keys` drops the rotating session keys immediately instead of waiting a few hours for their TTL to expire, which clears `AUTH_INSECURE_ROTATING_SERVICE_KEY_TYPE` right away. All service daemons must already run 20.2.4 binaries before you use it. Existing sessions are not affected, clients simply refresh their tickets at the next auth epoch. If you can tolerate the wait, skip this command and let TTL do the work, which is what the documentation recommends for most deployments.

Verify with three commands: `ceph --format=json mon dump | jq -r '.auth_service_cipher.name, .auth_preferred_cipher.name'` should print aes256k twice, `ceph health detail` should no longer list the rotating key warning, and `ceph --format=json auth dump-keys` shows the key type of every entity and rotating key, a command Patrick Donnelly added for exactly this check.

Note what this procedure does not do: it never touches `client.admin`, application identities, or kernel client keys. Those keep their aes keys and keep working, because aes stays in the allowed cipher list. Removing that list entry is the final step of the whole migration, and it gets its own section below.

Once every OSD is up and back in, remove the protective flags. Do not leave `noout` set longer than the operation: an OSD that dies for real while `noout` is on leaves its PGs degraded without triggering recovery, which is how a protection flag becomes an outage amplifier. The health mute expires on its own after two hours.

```
ceph osd unset noout
ceph osd unset norebalance
```

Rotating OSD keys on a cephadm cluster

The AUTH_INSECURE_SERVICE_KEY_TYPE count only drops when every OSD entity key is rotated to aes256k, and cephadm does none of it for you. The safe sequence runs host by host: enter maintenance, rotate each OSD key in the Monitor auth database, write the same secret into the BlueStore label, exit maintenance, and wait for active+clean before the next host. This is the loop that clears the warning, and it is what makes the `ceph mon set auth_service_cipher aes256k` step of the sequence above runnable on a cluster where the OSD keys are still aes.

For cephadm-managed OSDs there is no persistent on-disk keyring file for the daemon. The secret lives in the Monitor auth database and, once written, in the BlueStore label on the device. Rotating the key without updating the label produces an OSD whose label still carries the old aes secret, and at the next activation `ceph-volume` restores it, so the rotation and the label write belong in the same step. Two details keep the loop short: `ceph orch host maintenance enter` and `exit stop` and start every daemon on a host in one shot, and `readlink -f /var/lib/ceph/osd/ceph-<id>/block` resolves the device behind the block symlink for both LVM and raw OSDs, no `ceph-volume lvm list` lookup needed. Note that `set-label-key` accepts only `--dev`, never `--path`, and its `-v` takes the literal secret value, not a file path.

Three preparation steps come first. Cephadm administers the fleet with its own SSH identity, stored in the Monitor config-key store rather than in root's home directory, so retrieve it before the first hop. Pin an image variable to the cluster's own container image, and run every ceph command from inside `cephadm shell --image ${image}`: a bare-host ceph or ceph-authtool binary older than the cluster cannot parse an aes256k key at all, and once the migration has started that failure looks like a corrupt keyring file.

```
cephadm_user=$(ceph cephadm get-user)
ceph config-key get mgr/cephadm/ssh_identity_key > /tmp/cephadm_id_rsa
chmod 600 /tmp/cephadm_id_rsa
image=$(ceph config get mgr mgr/cephadm/container_image_base)
```

The loop itself, one host at a time with a real wait for active+clean between hosts:

```
wait_for_active_clean() {
    local timeout=${1:-1200} waited=0 not_clean
    while true; do
        # scrubbing is normal background activity; block only on states that are
        # not a superset of active+clean (degraded, undersized, peering, ...)
        not_clean=$(ceph -s -f json | jq '[.pgmap.pgs_by_state[] | select(.state_name |
(startswith("active+clean") | not))] | length')
        [[ "${not_clean}" -eq 0 ]] && { echo " active+clean (scrubbing ignored)."; return 0; }
        (( waited >= timeout )) && { echo " TIMEOUT after ${timeout}s, ${not_clean} PG group(s)
still not active+clean" >&2; return 1; }
        echo " waiting for active+clean (${not_clean} PG group(s) not clean, ${waited}s
elapsed)..."
        sleep 10
        waited=$(( waited + 10 ))
    done
}

hostlist=$(ceph orch ps --daemon-type osd -f json | jq -r '[] | .hostname' | sort -u)

for host in $hostlist; do
    echo "=== ${host} ==="
    ceph orch host ok-to-stop ${host} || continue
    ceph orch host maintenance enter ${host} || continue

    failed_osds=""
    for osdid in $(ceph orch ps --hostname=${host} --daemon-type osd -f json | jq -r
'[] | .daemon_id'); do
        echo "--- osd.${osdid} ---"
        newkey=$(ceph auth rotate --key-type=aes256k osd.${osdid} | grep -m1 '^[:space:]*key = '
| sed -E 's/^[[:space:]]*key = //' )
        label_written=0
        if [[ -n "${newkey}" ]]; then
            # maintenance enter does not guarantee the OSD container has released the
            # device yet (bluefs/rocksdb close can lag a few seconds); retry on busy

```

```

        for attempt in 1 2 3 4 5; do
            if ssh -i /tmp/cephadm_id_rsa -o StrictHostKeyChecking=accept-new
${cephadm_user}@${host} \
                "sudo cephadm --image ${image} shell --name osd.${osdid} -- bash -c 'dev=$(readlink
-f /var/lib/ceph/osd/ceph-${osdid}/block); ceph-bluestore-tool --dev \dev set-label-key
--key osd_key -v ${newkey}'; then
                    label_written=1
                    break
                fi
            echo " device busy, retrying osd.${osdid} in 5s (attempt ${attempt})..." >&2
            sleep 5
        done
    fi
    if [[ ${label_written} -eq 0 ]]; then
        echo " !! FAILED rotating osd.${osdid} (keyring/label mismatch), it will not come back
up on its own" >&2
        failed_osds="${failed_osds} ${osdid}"
    fi
done

if [[ -n "${failed_osds}" ]]; then
    echo "!! osd(s) left in a bad state on ${host}: ${failed_osds} - fix manually before
exiting maintenance" >&2
    break
fi

ceph orch host maintenance exit ${host} || { echo "!! maintenance exit failed on ${host}"
>&2; break; }

# maintenance exit should start every daemon on the host; confirm it did,
# since a failed rotation above leaves the OSD down for good
for osdid in $(ceph orch ps --hostname=${host} --daemon-type osd -f json | jq -r
'.[].daemon_id'); do
    status=$(ceph orch ps --hostname=${host} --daemon-type osd --daemon-id=${osdid} --refresh
-f json | jq -r '[0].status_desc')
    if [[ "${status}" != "running" ]]; then
        echo " osd.${osdid} is '${status}' after maintenance exit, forcing start" >&2
        ceph orch daemon start osd.${osdid}
    fi
done

ceph orch ps --hostname=${host} --daemon-type osd --refresh
wait_for_active_clean || break
done

```

Two details in the loop carry the safety argument. The grep and sed pair pulls the new secret straight out of ceph auth rotate's keyring output: ceph-authtool -n osd.<id> -p <file> looks like the natural tool for that, and on some builds it fails with error reading file on any input file, a fresh mktemp output included, even though the same file is valid keyring syntax. And the SSH hop into the OSD host runs from the admin shell, never from inside the per-daemon cephadm shell: that shell only carries the OSD's own, possibly broken, cephx identity, not admin access to the Monitors.

If you only need the warning gone quickly, updating the keyring alone is enough for authentication today, and skipping ceph-bluestore-tool looks tempting. On the next reboot or redeploy, ceph-volume re-activates the OSD from the on-disk label and silently restores the old aes key, so the warning comes back. Doing both writes in the same step costs nothing extra, which is how the loop above runs.

A rotation that half-failed is visible and recoverable. An OSD that starts against a rotated Monitor key with an old label dies with handle_auth_bad_method and Failed to start Ceph osd.<id>. The fix is to rewrite the label with the secret the Monitor now holds, then start the daemon again:

```

fsid=$(ceph fsid)
host=<hostname>

for osdid in $(ceph orch ps --hostname=${host} --daemon-type osd --refresh -f json | jq -r
'.[0].status_desc=="error" | .daemon_id'); do
    echo "--- osd.${osdid} ---"

```

```

key=$(ceph auth print-key osd.${osdid})
ssh -i /tmp/cephadm_id_rsa -o StrictHostKeyChecking=accept-new ${cephadm_user}@${host} \
    "sudo cephadm --image ${image} shell --name osd.${osdid} -- bash -c 'dev=$(readlink -f
/var/lib/ceph/osd/ceph-${osdid}/block); ceph-bluestore-tool --dev \dev set-label-key --key
osd_key -v ${key}'"
ssh -i /tmp/cephadm_id_rsa -o StrictHostKeyChecking=accept-new ${cephadm_user}@${host} \
    "sudo systemctl reset-failed ceph-${fsid}@osd.${osdid}.service"
ceph orch daemon start osd.${osdid}
done

ceph orch ps --hostname=${host} --daemon-type osd --refresh

```

One special case keeps the loop unblocked. An OSD whose disk already failed and whose replacement is deferred should not hold its host hostage. With the OSD down and out and the cluster active+clean, run `ceph orch osd rm <osdid> --replace`. It removes the daemon, clears `CEPHADM_FAILED_DAEMON`, and deletes the entity key from the auth database, which drops that OSD out of the `AUTH_INSECURE_SERVICE_KEY_TYPE` count, so the host's loop simply skips it. Because `auth_preferred_cipher` is already `aes256k` by then, the replacement OSD comes up on the secure cipher with no extra rotation.

Once the loop has crossed every host, `ceph health detail` no longer lists the service-entity warning, and the cipher and cleanup steps of the post-patch sequence above can run in order.

Read the authentication health warnings by what they measure

`AUTH_INSECURE_SERVICE_KEY_TYPE` means a service such as an OSD is still authenticating to the Monitors with an aes key. It clears when that entity key is rotated.

`AUTH_INSECURE_ROTATING_SERVICE_KEY_TYPE` is different: it concerns the short lived rotating service keys, and it is the one warning that resolves itself. [Eugen Block spells this out in the thread](#): once the Monitors are patched and `auth_service_cipher` is set to `aes256k`, new rotating keys are created with the new cipher and the legacy aes rotating keys expire after a few hours on the default TTL, with no action required. Wiping the rotating key database only clears the warning sooner, and the documentation does not recommend it for most deployments. Do not confuse this self-expiring layer with the entity keys: those still need the rotation procedure.

Remove the legacy cipher last

```
ceph mon set auth_allowed_ciphers aes256k
```

This command belongs at the very end of the whole migration, after every client key is migrated. The moment you run it, any entity still holding an aes key stops authenticating: kernel clients, copies of `client.admin` on other nodes, applications with their own identities. It also flips the runtime default of `mon_auth_allow_insecure_key` back to `false` and clears the `AUTH_INSECURE_KEYS_CREATABLE` warning. Run it when nothing needs aes anymore, not as part of the patch day.

Treat RBD and virtualization as an application migration

For RBD clients behind hypervisor or cloud orchestration, a key rollover can touch libvirt secrets, VM templates, live migration, and the control plane. Community replies show that these details vary by platform. Do not assume that replacing a secret changes the Ceph username consumed by every running workload. Build a node-by-node cutover plan, test a representative migration or restart path, and keep a rollback credential until the new identity works in production.



Package installation is the first stage. Manual daemon-key rotation and client-key rotation plus compatibility checks complete the remediation. None of it is automated.

Plan the maintenance window around key rotation and client compatibility, not package installation alone. Nothing in 20.2.4 rotates OSD and MDS keys for you, so the tested manual procedure is part of the window, not an add-on.

The same month: French public-sector disclosures

French authorities disclosed several public-sector security incidents the same month Ceph shipped these fixes: DGFIP, FICOBA, France Titres / ANTS, and the Ministry of Education. I list them as operational context, not because any involves Ceph. No source reviewed connects Ceph, or any of these four CVEs, to an incident below.

DGFIP, the French tax administration. Intrusions in June and July 2026 used stolen credentials from an agent and an authorized third party. Data relating to 678,000 individuals and businesses was consulted and extracted. DGFIP states that taxpayer portals and passwords were not compromised. Read the [Finance Ministry press release](#) and the [DGFIP notice](#).

FICOBA, the national bank-account register. In early 2026, impersonation of an authorized official's credentials allowed unauthorized queries exposing information on roughly 1.2 million bank accounts. The accounts themselves were not accessed. Read the [DGFIP FICOBA notice](#).

France Titres / ANTS. A serious incident potentially involved up to 11.7 million portal accounts. The Interior Ministry says it detected the incident on April 15, 2026. Read the [Interior Ministry update](#).

Ministry of Education. The ministry confirmed an intrusion on July 25, 2026, with possible exfiltration of personnel data. The scope was still under investigation. Headline claims circulating online remained unverified. Read the [Education Ministry notice](#).

DGFIP and FICOBA specifically involved stolen or impersonated credentials. The official France Titres and Education notices cited here do not establish the same attack vector. Together, these incidents motivate identity, least-privilege, and detection controls, but they do not prove one common failure pattern. None of the reviewed official notices attributes an incident to a storage-layer vulnerability.

For scale, [ANSSI handled 3,586 security events in 2025](#). [CNIL received 6,167 breach notifications that year](#), half involving hacking.

What this means for your storage platform

A patch release fixes four bugs. It does not make a platform resilient by itself. The Ceph upgrade remediates four specific vulnerabilities, but it is not a substitute for key management, least privilege, monitoring, tested recovery, and incident response. If you operate Ceph, MinIO, or another S3-compatible store, use this checklist:

1. **Identity first.** Rotate keys and credentials on a schedule. Scope every CephX identity and every RGW or S3 credential to the minimum capability. An identity with `mon allow *r` should never be able to read your secret store.
2. **Patch discipline.** Track upstream security releases for your storage layer as closely as kernel CVEs. This release requires planning, so budget the window now.
3. **Tested backups.** An untested backup is not evidence of recoverability. Measure restore throughput and validate recovery ordering. Replication is not a backup because deletes and corruption propagate.
4. **Least privilege at every boundary.** Presigned URLs, STS tokens, and service accounts each delegate trust. Use short expiries where supported for delegated credentials. Define scope and rotation for service accounts.
5. **Monitoring for anomalous access.** Look for mass metadata reads, unusual token use, and queries outside normal business patterns. The disclosed record counts show why early detection matters, but the cited notices do not quantify how detection delay affected their scope.
6. **Incident readiness.** Decide who rotates each credential, who assesses notification duties and contacts CNIL when required, and who can take systems offline. DGFIP's disclosures show why the response plan matters once access occurs.

Where to start

Start with the [release announcement](#) and schedule the upgrade with the key-rotation steps included. If you run Ceph in production, read it today and plan the maintenance window around key rotation and client compatibility, not package installation alone. The upgrade remediates four specific vulnerabilities, and it is not a substitute for key management, least privilege, monitoring, tested recovery, and incident response.

Then assess the surrounding controls. My [Resilience and Disaster Recovery Assessment](#) reviews measured restores, replication behavior, access controls, and your actual RTO and RPO. [Book a free 15-minute introductory call](#) to see whether it fits your platform.

Update 2026-09-29. This post is now part of [Ceph in Production: RBD, RGW and CephFS](#), the field guide to Ceph in production.

Related posts

- [Upgrading a Ceph multi-site S3 platform from Squid to Tentacle](#) wp:list-item
- [Ceph in Production: RBD, RGW and CephFS](#)
- [Migrating unmanaged OSDs to managed OSDs in Ceph Squid](#)
- [Ceph RBD mirroring for Kubernetes storage in production](#)

PDF snapshot of the public article. The online version is authoritative; links remain clickable.