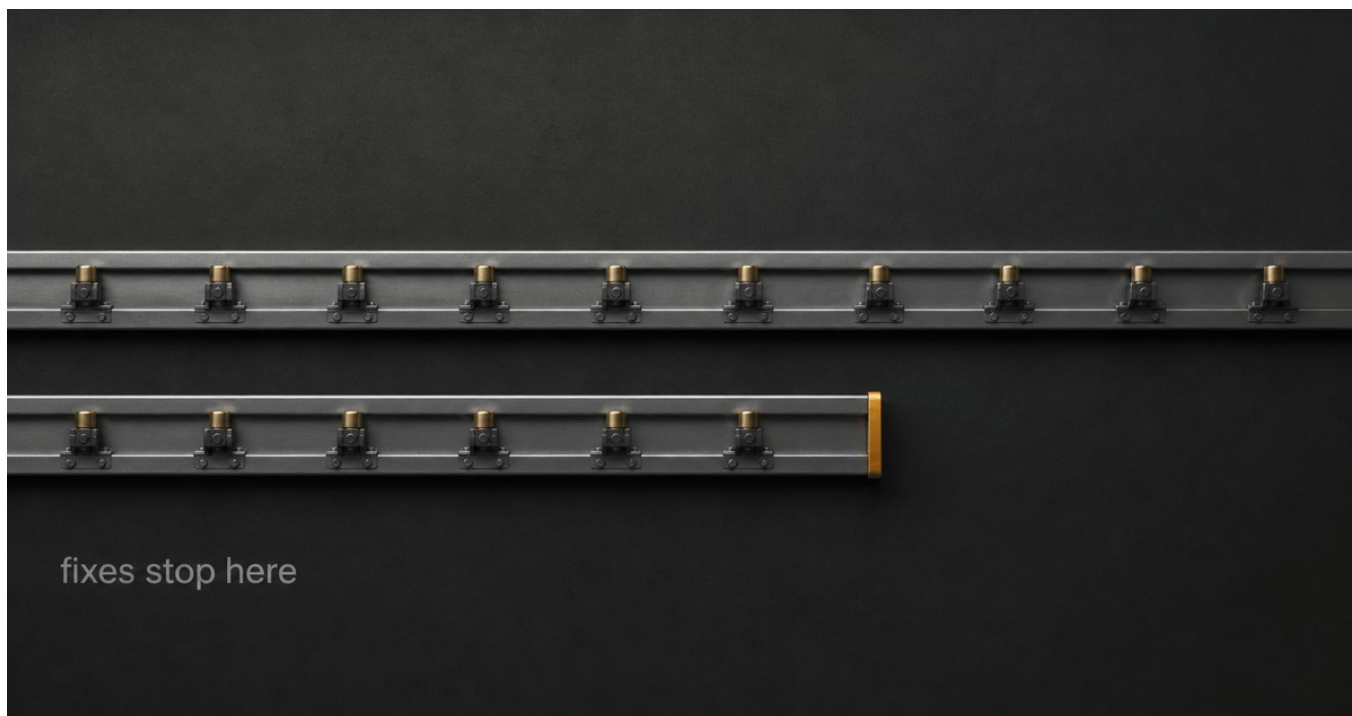


MinIO Community in Production: Assess the Risk You Are Running

Julien Laurenceau | Published 2026-09-14 | Updated 2026-09-30

<https://pepitedata.com/minio-community-edition-production-risk/>

PepiteData | LinkedIn: Julien Laurenceau | X: @Julien_Lau



Update, September 2026: the fork contribution mentioned below now has its own write-up. The three retry-side defects behind the storm, the design call my own suggestion got wrong, and the convergence tests: [how the Silo fork fixed the 405 replication loop](#). If you are weighing Community, the fork or AIStor for a production data lake, that release-channel assessment is part of what I run: [MinIO, S3 and object storage consulting](#).

If you run the Community edition of MinIO today, you are running a release that has stopped moving. The last publicly available open-source release, `RELEASE.2025-10-15T17-29-55Z`, is still the latest one the GitHub releases API returns, and the `minio/minio` repository itself went read-only in April 2026. MinIO did not stop developing the software. Development moved to AIStor, the vendor's closed-source product line, whose release notes stay public at `d1.min.io` even though the code does not.

Those notes are the only remaining public signal of what is being fixed in the codebase your Community deployment shares its lineage with. So while auditing stacks that run Community MinIO, I have taken to reading another product's changelog. Not for curiosity: before a scale-up, a hardware purchase or a vendor decision, the release channel is part of the platform risk, and right now it is the part nobody priced in.

This post walks through the AIStor fixes that plausibly concern Community operators, grouped by what they threaten, ranked by how sure I am that they apply, and with the evidence level stated on every line. A note checked on 2026-08-26 against the live release-notes page is the source for everything below; where I mark something as inference, it is inference.

The short version

The short version: Community MinIO stopped receiving releases in October 2025 and the repository went read-only in April 2026, while development continued as the closed-source AIStor. Three CVEs are confirmed to affect the final Community release and none will be patched there. A maintained fork, PGSTY Silo, now ships patched open-source builds. Everything else below is inference from shared code paths.

- The last MinIO open-source release is `RELEASE.2025-10-15T17-29-55Z`, dated 15 October 2025. No newer one exists.
- The `minio/minio` repository was archived read-only in April 2026. Community MinIO is source-only, with no pre-compiled releases.
- Development continues in AIStor, MinIO's closed-source product. Its release notes stay public at `dl.min.io`; its code does not.
- Three CVEs carry public GitHub advisories stating they affect the open-source line through its final release: CVE-2026-34204 (High), CVE-2026-33322 and CVE-2026-33419 (both Moderate). None of the three will be patched there.
- A maintained community fork now exists: PGSTY Silo continues the open-source server with releases, packages, images and the full web console, and it has already shipped fixes for all three CVEs above (checked 2026-09-14). It is a maintained alternative to evaluate, not upstream MinIO resuming.
- Everything else here is inference from shared code paths, and every item says so on its own line. An AIStor fix is evidence of a defect class, not proof about your binary.

Is MinIO Community Edition still maintained?

No. The `minio/minio` repository is archived and read-only, no open-source release has shipped since `RELEASE.2025-10-15T17-29-55Z`, and there is no published plan to resume one. The MinIO server itself is still being developed, under a different name and a different license.

AIStor is MinIO's commercial, closed-source distribution of the MinIO server. It is not a fork in the sense of a diverging rewrite: it is the same vendor continuing the same core server, with the development artifacts now kept out of public view. The public open-source releases ended in October 2025. That distinction matters for how much weight you can put on the notes.

Because the two lines share code, a defect class fixed in AIStor usually says something about the shared code, but not always. Some fixes repair regressions introduced after the split, in code the Community release never contained. And the AIStor issue references (#6711 and friends) are internal tracker IDs that are not resolvable publicly, so no one outside MinIO can diff an AIStor fix against the archived Community repo. An AIStor fix is therefore evidence of a defect class, never proof that your exact Community binary carries it. Only the four fixes with public GitHub security advisories carry vendor-stated applicability, and of those four, three apply to the final Community release; the fourth was already patched in it.

Which MinIO CVEs are confirmed to affect the Community edition?

Three CVEs are confirmed to affect the Community line, because their public advisories on the archived GitHub repository state it explicitly. All three say the affected versions run "through the final release of the `minio/minio` open-source project". A fourth, a service-account privilege escalation, was already fixed in the October 2025 release itself, so it does not apply.

Fix	Affected behavior	Impact	Evidence	AIStor fix	Workaround
CVE-2026-34204 (GHSA-3rh2-v3gr-35p9, High)	Any user with <code>s3:PutObject</code> can inject internal SSE metadata through <code>X-Minio-Replication-*</code> headers	Uploaded objects become permanently unreadable	Confirmed: public advisory names the Community line	RELEASE.2026-03-26	Reverse-proxy rule dropping the injection headers; documented in the advisory
CVE-2026-33322 (GHSA-5cx5-wh4m-82fh, Moderate)	JWT algorithm confusion in OIDC login: knowing the OIDC <code>ClientSecret</code> lets an attacker forge tokens, up to <code>consoleAdmin</code>	Full impersonation on OIDC deployments; requires the <code>ClientSecret</code>	Confirmed: public advisory names the Community line	RELEASE.2026-03-17	Treat the <code>ClientSecret</code> as critical infrastructure; no Community patch
CVE-2026-33419 (GHSA-jv87-32hw-hh99, Moderate)	No rate limiting on <code>AssumeRoleWithLDAPIdentity</code> ; username enumeration and unauthenticated brute force	Credential guessing on LDAP deployments exposed to untrusted networks	Confirmed: public advisory names the Community line	RELEASE.2026-03-17	Proxy rate-limiting of the STS endpoint, firewall restriction, LDAP lockout (with a DoS tradeoff)

The first one deserves emphasis. CVE-2026-34204 is a data-availability issue, not an information disclosure: an ordinary writer, not an admin, can make objects permanently unreadable, one by one or bucket-wide. If your Community deployment accepts `PutObject` from anything beyond fully trusted internal callers, the documented reverse-proxy workaround is not optional.

The severity labels also deserve one caveat: the vendor's AIStor notes used Critical, CVSS 9.x language for the OIDC and LDAP items, while the public GitHub advisories list them as Moderate. The GHSA severities are the public record; the exploitability genuinely depends on your deployment.

Which AIStor fixes probably apply to your Community build?

Everything in this section is inference from code-path overlap. I rank these by user-visible risk, and I flag the confidence honestly: the failure modes are version-independent classes, but whether the Community binary contains each specific bug is unverifiable from public sources.

Key rotation and IAM corruption (AIStor 2026-08-07, internal #6711). The AIStor note itself calls this the headline reason to upgrade. Re-encrypting IAM and configuration stores during key rotation mixed an encryption context into the authenticated data, leaving those stores permanently undecryptable after a rotation with a non-empty context. The job reported success; the failure surfaced at the next restart. Data-loss class, silent failure, and the closest analogue to the replication silent-drop problems I have written about before. If the Community key-rotation path shares this logic, running key rotation on the October 2025 release could brick config decryption. That is a conditional sentence on purpose: I cannot confirm it. My operational suggestion, and mine only, is to avoid key rotation with a non-empty context until you have verified your build.

Erasure-set write integrity (AIStor 2026-03-12, internal #3264, #3081, #3133, #3200). A cluster of fixes around partial-quorum writes: metadata left inconsistent when a subset of drives errored, stale data overwritten by cleanup paths, reader data races, and a panic on pre-2023 metadata. Long-lived Community deployments have old objects and occasional drive failures; the failure modes are version-independent even if the specific defects are not. Mitigation is operational: keep full write quorum, avoid `CopyObject` and `HeadObject` storms against the same bucket, alert on panic loops.

S3 Select crash class (AISTor 2026-05-28, #4665, #4951). An output-buffer bug present since 2020 could crash the server mid-query; one reported deployment logged fifteen-plus restarts in two days, and a related zip-handler fix addressed a Content-Length overflow. Age matters here in the useful direction: a defect that old predates the Community freeze, so the inference is about as strong as inference gets without a diff. Detection is cheap: watch for Select-induced restarts; avoidance cheaper still: do not point untrusted input at S3 Select.

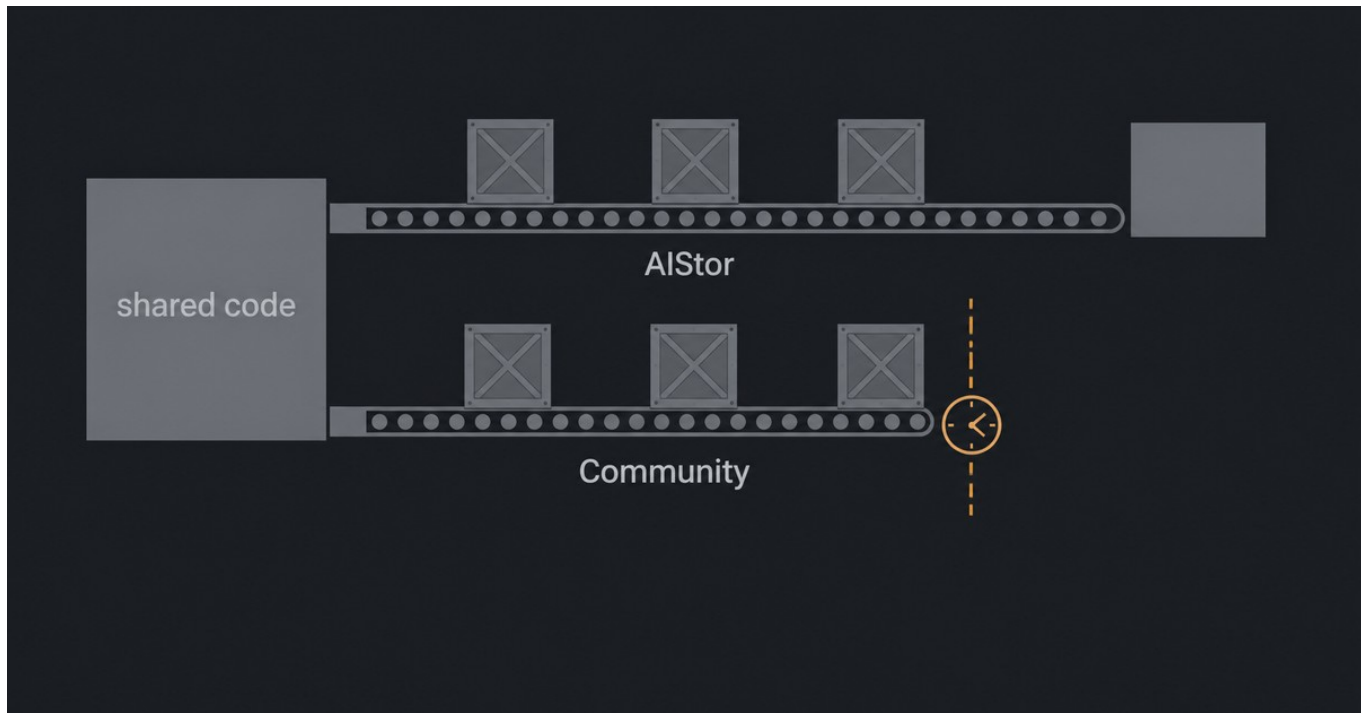
Bitrot read corruption on unaligned shards (AISTor 2026-03-26, #3522). Intermittent "file is corrupted" GET errors for objects whose bitrot file size both exceeds the erasure block (1 MiB) and is not a 4096-byte multiple, most visible on EC:1 sets. Detection is easy: alert on that error string. There is no documented workaround.

Availability: pool expansion and full disks (AISTor 2026-03-20, #3423, #3520). Buckets invisible after adding a pool, because quorum was computed across pools instead of per pool; and a single full disk, space or inodes, producing cluster-wide HTTP 507s. Both code paths are core. The mitigation is mundane and effective: monitor per-drive space and inodes, and verify bucket listings after any pool change rather than trusting the CLI output.

Writes misreported as 404 (AISTor 2026-05-04, #4206). A write-quorum failure returned 404 NoSuchKey instead of 503, so well-behaved clients did not retry. In a replication pipeline, that converts a transient failure into apparent object loss. The staged-metadata race involved is core commit-path code, which is why I rank the inference here as strong.

A ten-month backlog of auth hardening (AISTor 2026-03 through 2026-08, about a dozen items). Unauthenticated log-streaming endpoints, an OpenID audience-check bypass, IP-allowlist bypass via X-Forwarded-For, non-expiring STS credentials, policy-evaluation bypasses via object tags, path traversal in the internal storage REST layer. Individually each is "strongly indicated to uncertain"; collectively they say the Community line accumulated roughly thirteen policy-evaluation and auth-edge hardening fixes it never received. I am not going to claim any of these is exploitable on your build: the fact-check standard here is that absence of a public fix is not proof of an open hole. The generic mitigation stands regardless: isolate the admin and STS endpoints at the network layer, which costs nothing and caps all of these classes at once.

Replication and ILM interactions (AISTor 2026-03-12 #3201, 2026-05-04 #4210). ILM expiring objects whose replication had not completed, because the scanner cleared replication config on a transient failure; and phantom delete markers from Hadoop-style DROP PARTITION PURGE operations, 404ing objects on both clusters. Of the two, only the ILM mechanism has a plausibly version-independent failure path; the delete-marker case rides on a narrower trigger combination, so treat it as uncertain rather than strongly indicated. These are among the highest data-loss-adjacent items for anyone running replication with lifecycle rules, which, given how much of my MinIO work involves replication-based DR, is the population I most worry about. Inference, not confirmation.



Two release lanes from one codebase. The AIStor lane keeps receiving fixes after October 2025; the Community lane receives none. Shared defect classes persist on the stalled lane until independently verified.

How to assess and reduce the risk

Assess and reduce the risk in four actions, in order of return on effort. Verify your exposure to the three confirmed CVEs first, because the SSE-header injection has a documented proxy workaround that is not optional if exposure exists. Then freeze the risky operations, test the failure modes instead of assuming them, and decide the release-channel question deliberately, not on a deadline panic.

1. **Verify your exposure to the three confirmed CVEs.** Check whether untrusted callers can PutObject (the SSE-header injection), where your OIDC ClientSecret lives, and whether the STS LDAP endpoint touches an untrusted network. The first has a documented proxy workaround; apply it if exposure exists.
2. **Freeze the risky operations.** Hold off key rotation with a non-empty encryption context and treat pool expansions as change-window events with post-checks, until each is verified against your build.
3. **Test the failure modes instead of assuming them.** The items above cluster in exactly the places a DR audit exercises anyway: replication lag under partition, restore throughput, bucket visibility after topology change. If you have not measured your RTO and RPO recently, the frozen release channel is one more reason the claimed numbers are not the real ones ([RTO/RPO: claimed vs measured](#)).
4. **Decide the release-channel question deliberately.** Not emotionally, and not on a deadline panic. Whether to stay on a frozen Community line, move to a maintained fork, move to AIStor, or move to another object store is a vendor and budget decision that deserves a scoped evaluation, before the next scale-up or hardware purchase makes it for you.

Method: how I mapped AIStor fixes to the Community release

I read the public AIStor release notes from RELEASE.2025-10-17 onward (checked 2026-08-26), classified each fix by subsystem, and mapped each to the Community release by code-path overlap. Confidence labels: *confirmed* means a public advisory explicitly names the Community line; *strongly*

indicated means the fix touches core code demonstrably present in the Community tree, with no public advisory; *uncertain* means I cannot rule out that the bug lived only in post-split code.

I excluded AIStor-only subsystems (RDMA builds, Tables/Iceberg, Delta Sharing, licensing) and fixes for regressions introduced after October 2025. Two further exclusions: regressions the notes themselves date after October 2025 in AIStor-only code, and the toolchain bump set, which is better handled by running `govulncheck` against your own binary than by my inference.

The honest limit: only MinIO staff, or a diff of the archived Community repository against the internal tracker IDs, can turn "strongly indicated" into "confirmed". Until someone does, treat the confirmed table as your action list and the rest as a prioritized verification backlog, not as a list of open holes in your deployment.

A maintained fork now exists: PGSTY Silo

A maintained fork now exists: [PGSTY Silo](#), from the team behind the Pigsty PostgreSQL distribution, continues the open-source MinIO server. The release-channel landscape has changed since I finished the AIStor mapping above. It ships what Community lost: regular releases, RPM, DEB and APK packages, multi-arch container images, a Helm chart, and the full web console that upstream cut back to a stub.

The lineage has shipped releases since December 2025 as `pgsty/minio` and was renamed to `pgsty/silo` in August 2026. The license stays AGPL-3.0 with no CLA: contributions are accepted inbound and outbound under the same terms, with DCO sign-off as the only requirement. Pigsty runs it in production as its PostgreSQL backup repository.

The compatibility contract is the part an operator should read first. The S3 API, the `MINIO_*` environment variables, the `x-minio-*` headers, the `/minio/*` routes, the metric names and the on-disk format, including `.minio.sys`, are preserved and held in place by a CI compatibility check. What changes is only the delivery surface: the executable, the package, the service, the chart and the image are named `silo`, and no `minio` binary alias is installed. Their compatibility audit lists every divergence from upstream in code-verified form.

On security, the public record answers the confirmed-CVE table above: Silo has shipped fixes for all three CVEs that affect the final Community release. CVE-2026-34204 and CVE-2026-33322 were fixed in `RELEASE.2026-04-17`, and CVE-2026-33419 received its complete fix in `RELEASE.2026-06-18` after two correction rounds, each documented in a public advisory on the [security chronicle](#). The fork also runs its own hardening series, five items shipped in the 2026-09-03 release alone. One nuance worth adopting from their practice: a published release can trail the main branch, so read the advisory status page instead of assuming the newest release carries every fix.

The fork also accepts contributions, which is where this post becomes personal. The delete-marker convergence loop behind [the replication storm I documented earlier this month](#) lives in code the fork inherited, and it was reported against the fork. The maintainer merged a first fix on 2026-09-09: marker purges classified like the other delete paths, MRF queue drops made observable, resync cancellation made reliable. I opened [pull request #184](#) on 2026-09-14 to complete that convergence work on the retry path. A purge whose first delivery failed was never retried, because the early-out tested the marker's creation status. Purge outcomes were recorded in the wrong status field, so the source stayed PENDING forever. And the fastest retry path dropped every delete-marker entry, because an info lookup on a marker version answers 405. Those stuck states are the fuel of the storm: thousands of requests per second where 96.8 percent of the answers carried no data. The pull request is open and under review, not merged, so I describe it as a proposal, not a shipped fix.

What the fork does not remove is the evaluation work. It is a young project carried by a small team, and their own guidance is to treat each release as a downstream upgrade: pin versions, read the release

notes, keep a rollback path. For a production cluster, this is not MinIO resuming. It is a fourth option in the release-channel decision, with a real maintenance story and one more dependency to manage deliberately.

Should you stay on MinIO Community Edition?

Running Community MinIO in 2026 is a legitimate choice for plenty of workloads. What is not legitimate is running it while assuming it still receives fixes. The frozen line carries three confirmed, unpatched CVEs, and a plausible, unproven, backlog of integrity and hardening fixes behind them. The AIStor release notes remain the vendor's public window into that development; the maintained Silo fork is the second window, and the only one that ships patched open-source builds.

Which of the two you pick, or neither, is a vendor and budget decision. What is not defensible is running the frozen line by default.

If you want a second pair of eyes on your object-storage layer, release channel, replication design, DR numbers you can defend, I do that as a short engagement. Book an intro call at cal.com/julien-pepitedata/15min and we will figure out which one fits, or start from the [data platform performance audit](#) page.

Frequently asked questions

Is MinIO Community Edition still receiving security updates?

No. The final open-source release is RELEASE.2025-10-15T17-29-55Z and the repository has been archived read-only since April 2026. Three CVEs disclosed after that date have public advisories naming the open-source line as affected, and none of them has an open-source patch. In practice the Community edition reached end of life for security purposes in October 2025, whether or not the vendor uses that phrase.

Which CVEs affect the last MinIO open-source release?

CVE-2026-34204 (SSE metadata injection through X-Minio-Replication-* headers, High, has a reverse-proxy workaround), CVE-2026-33322 (JWT algorithm confusion in OIDC login, Moderate) and CVE-2026-33419 (no rate limiting on AssumeRoleWithLDAPIdentity, Moderate). A fourth advisory, a service-account privilege escalation, was already fixed inside the October 2025 release and does not apply.

Can I patch Community MinIO myself?

Technically yes: the source of the final release is still readable and AGPL-licensed, so you can carry a patch and build your own binary. From that point you own the build, the testing and the distribution, permanently, for a codebase you did not write. The organized version of that path now exists: the Silo fork above maintains patched builds, packages and images, so you can consume a maintained release instead of carrying your own. The tradeoff is a new dependency on a young project. For most teams the cheaper controls remain the network-layer ones in this post: drop the injection headers at the proxy, rate-limit the STS endpoint, and keep the admin surface off untrusted networks.



Pending fixes stack up behind a stopped Community lane while the supported lane keeps moving.

Is there a maintained fork of MinIO Community Edition?

Yes. PGSTY Silo, maintained by the team behind the Pigsty PostgreSQL distribution, continues the open-source MinIO server under AGPL-3.0 with no CLA. It ships regular releases, RPM, DEB and APK packages, multi-arch container images, the full web console, a CI-enforced compatibility contract on the S3 API and the on-disk format, and a public security advisory for every fix. It has already shipped fixes for the three CVEs that affect the final Community release. It is young and carried by a small team, so treat it as a new dependency to evaluate, not as upstream resuming.

How do I tell whether my deployment is exposed?

Three checks answer most of it. Can any caller you do not fully trust issue PutObject? Where does your OIDC ClientSecret live, and who can read it? Does the STS LDAP endpoint face a network you do not control? A no to all three drops your exposure to the confirmed set close to zero and leaves only the inferred items, which are a verification backlog rather than an incident.

Should I migrate from Community MinIO to AIStor?

That is a budget and vendor decision, not a technical one, and it deserves to be made before a deadline forces it. AIStor buys you the vendor's fix stream and support; the maintained fork buys you patched open-source builds without a commercial dependency. AIStor also means a commercial dependency on a vendor that has already demonstrated it will close a code line. The alternatives are staying on the frozen release with compensating controls, moving to the maintained community fork, or moving to another S3-compatible store. All of them are defensible; what is not defensible is picking by default.

Sources

- AIStor release notes index: dl.min.io/aistor/minio/release/notes/ (all 57 note files retrieved 2026-08-26)
- GHSA-3rh2-v3gr-35p9 / CVE-2026-34204: [GitHub advisory](#)
- GHSA-5cx5-wh4m-82fh / CVE-2026-33322: [GitHub advisory](#)

- GHSA-jv87-32hw-hh99 / CVE-2026-33419: [GitHub advisory](#)
- Final Community release: [RELEASE.2025-10-15T17-29-55Z](#)
- PGSTY Silo repository and compatibility audit: [github.com/pgsty/silo](#), [silo.pgsty.com/compatibility/server/](#) (checked 2026-09-14)
- Silo security chronicle, including the fix releases for CVE-2026-34204, CVE-2026-33322 and CVE-2026-33419: [silo.pgsty.com/blog/security/](#)
- Delete-marker convergence on the fork: [pgsty/silo#162](#) (merged 2026-09-09), [pgsty/silo#184](#) (open, under review)

Update 2026-09-29. This post is now part of [Object Storage for Data Platforms at Scale](#), the field guide to running object storage at scale.

Related posts

- [Object Storage for Data Platforms at Scale](#)
- [MinIO site replication bug: how 684 million HTTP 405s exposed the delete-marker loop](#)
- [MinIO Site Replication: Sync Mode Will Not Save Your RPO](#)
- [MinIO on XFS: Inode Exhaustion and Prefix Design for Lots of Small Files](#)
- [MinIO and Small Files: When Erasure Coding Becomes 15x Replication](#)

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