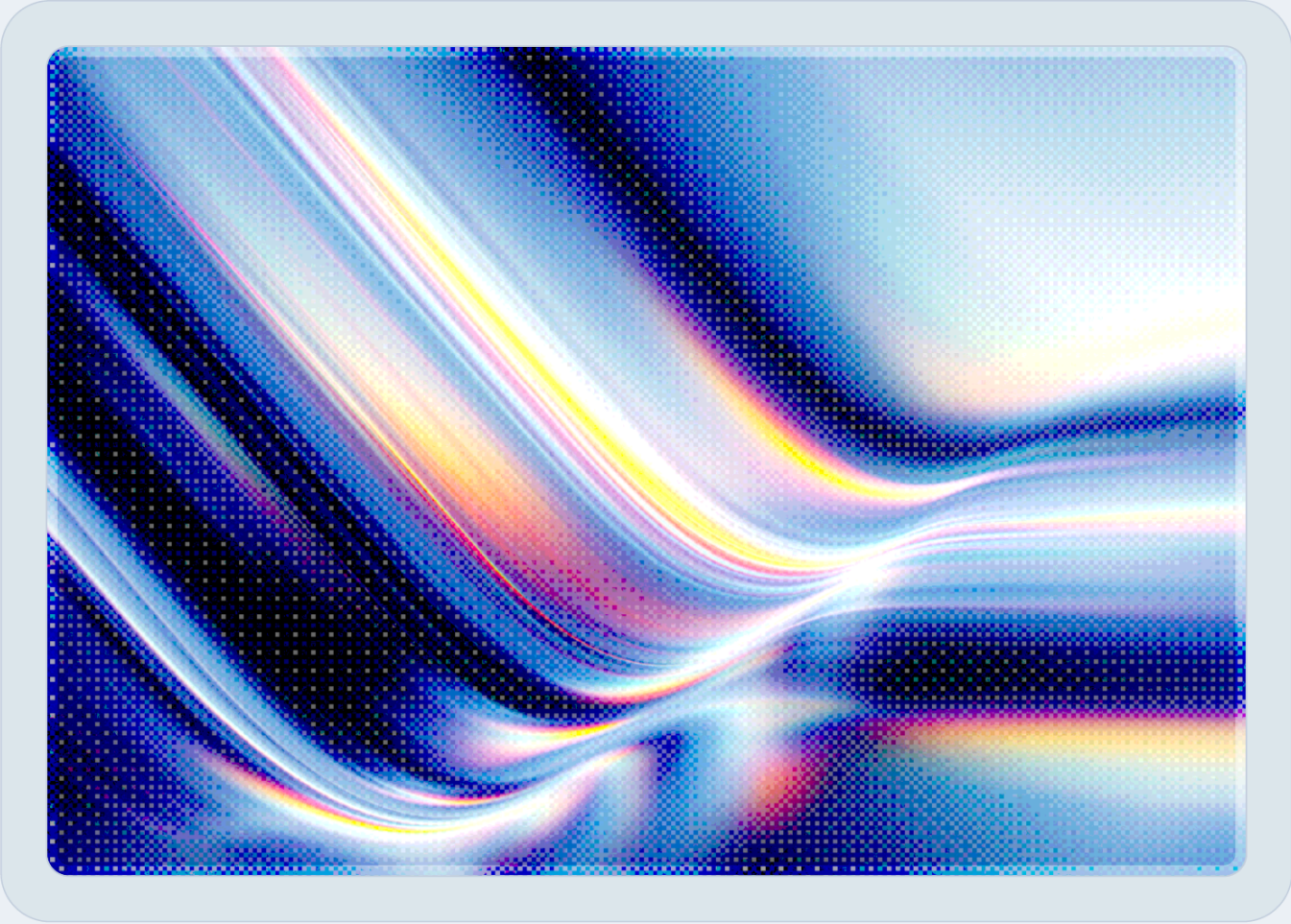
CSAI Foundation | Cloud Security Alliance

Metabase Zero-Day: Unauthenticated SQLi to Admin Takeover

2026-08-10

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

A maximum-severity, unauthenticated SQL injection flaw in Metabase — the widely deployed open-source business intelligence platform — was exploited in the wild before the vendor became aware of it, allowing remote attackers with no credentials to seize full administrator control of affected instances. Tracked by GitHub as GHSA-vwf4-m7j8-wcjf and scored CVSS 10.0, the vulnerability lives in the `/api/session/reset_password` endpoint and lets an attacker inject arbitrary SQL into the application database, ultimately granting the same privileges as a legitimate administrator [1][2][3]. Metabase discovered the campaign only after detecting anomalous activity against its own Cloud infrastructure, indicating the attackers had operational knowledge of the flaw before any public advisory existed [4][5]. Every release line from version 58 onward was vulnerable, spanning six branches of the product across roughly a year of releases, which suggests, though exact adoption figures are not published, that a meaningful portion of self-hosted deployments running anything but the newest patch release were exposed for an extended period [3]. Laptop maker Framework and form-builder vendor Tally have both confirmed that attackers used this flaw to steal customer data from their Metabase-connected environments [6][7]. Because administrator access inside Metabase exposes the credentials for every warehouse, database, or data lake the instance connects to, the practical blast radius of this bug extends well beyond the BI tool itself and into whatever analytical data estate it was built to query.

Background

Metabase is an open-source business intelligence and analytics application used by organizations to build dashboards, run ad hoc queries, and expose curated data views to non-technical staff without giving them direct database access. It is commonly self-hosted by smaller organizations because it is easy to stand up, connects natively to a wide range of production data warehouses, and can be deployed quickly for internal reporting against a SaaS product's backend databases. That convenience is also what makes a full-administrator compromise of a Metabase instance disproportionately damaging: the application's core function is to hold live, working credentials to whatever data sources an organization has connected to it, and an administrator account can read, export, or reconfigure all of them.

The vulnerability at the center of this incident sits in the password-reset flow, an endpoint that by design must be reachable without a prior login, since a locked-out user cannot authenticate before resetting a forgotten password. Metabase's advisory states that a remote, unauthenticated attacker could send

crafted input to `POST /api/session/reset_password` and have it interpreted as SQL against the application's own backing database, rather than as the token-validation logic the endpoint was meant to execute [2][3]. Successful exploitation let the attacker manipulate that backend query to grant themselves administrator-level access to the instance, at which point ordinary post-authentication functionality — user management, database connection configuration, dashboard and query export — became available to someone who had never presented a password [3][4].

Metabase has not attributed initial discovery to an external researcher or bug-bounty submission. Instead, the company says it identified the attack pattern by observing exploitation attempts against its own Metabase Cloud infrastructure, meaning the flaw was a true zero-day: it was being actively weaponized in the field before Metabase itself knew it existed [4][5]. Reporting places the earliest confirmed exploitation on or around August 3, 2026, with Metabase blocking the abused endpoint and patching its Cloud fleet within roughly three days, followed by a public security advisory and downloadable patches for self-hosted customers on August 6 [6][7]. No CVE identifier had been assigned to the flaw at the time of disclosure, which matters operationally: vulnerability scanners and asset-management tools that key off the National Vulnerability Database rather than vendor or GitHub Security Advisory feeds will not automatically flag exposed Metabase instances as vulnerable [1][8].

Security Analysis

The flaw's CVSS 10.0 rating reflects the convergence of several worst-case properties in a single bug: it requires no authentication, no user interaction, and low attacker sophistication, while yielding a complete compromise of confidentiality, integrity, and availability for the affected system, per the published vector `AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H` [3]. In practical terms, any internet-reachable Metabase instance running an affected version was exploitable by a single unauthenticated HTTP request, with identity-based controls such as MFA or account-lockout policies providing no friction at all, since those mechanisms only engage after an identity has been asserted and this attack path never asserts one; only network-layer controls such as a WAF or IP allow-listing could interpose.

The affected version matrix spans six release branches and roughly a year of development — broader than the version window typically affected by a single vulnerability of this severity. Metabase confirmed that every release from 58.0 through the pre-patch builds of 63.x carried the flaw, with versions prior to 58 unaffected. The table below summarizes the vulnerable ranges and their corresponding fixes, expressed per the GitHub Security Advisory's own exclusive-upper-bound notation (a listed vulnerable version is the last version affected; the version immediately after it may still be pre-patch and is covered by the same fix) [3][8].

Release branch	Vulnerable range	Patched version
58.x	58.0 – 58.22	58.24
59.x	59.0 – 59.19	59.21
60.x	60.0 – 60.15	60.17
61.x	61.0 – 61.9	61.11
62.x	62.0 – 62.7	62.9
63.x	63.0 – 63.2	63.5

Because the flaw spans roughly a year's worth of release branches, organizations running anything but the very latest patch releases across six separate version lines were exposed, regardless of how recently they had last upgraded. This pattern is consistent with a broader dynamic security teams should watch for: bugs introduced early in a shared code path — here, the password-reset handler — can persist silently across many subsequent releases unless specifically re-audited, since routine feature development and regression testing are not designed to catch injection flaws in rarely-exercised authentication edge cases.

Post-exploitation, the attacker's administrator session provided access to Metabase's stored connection credentials for every data source wired into the instance — production databases, cloud data warehouses, and any other system an administrator had configured for reporting. Metabase's own guidance explicitly warns that a compromised instance can be used to alter application configuration, harvest connected-database credentials, read any data reachable through those connections, and export it in bulk [2][3]. This is precisely the pattern that has now played out at two named organizations. Framework, the modular-laptop manufacturer, disclosed that attackers using this flaw accessed customer names, email addresses, login IP addresses, billing and shipping addresses, and phone numbers, with Framework for Business customers additionally exposed to VAT numbers, EIN data, and billing email addresses [6][7]. Tally, a form-building SaaS vendor, confirmed a separate compromise on the same date — August 3 — that exposed customer email addresses and password hashes, though Tally noted the hashes were one-way cryptographic values and that user-submitted form content, stored in a separate system, was not affected [6][7].

The published indicator of compromise for this campaign is a specific two-step HTTP pattern: a `POST` request to `/api/session/reset_password` that returns an HTTP 400 status, immediately followed by a `GET` request to `/api/user/current` that returns HTTP 200 [1][2]. The 400

response is notable because it indicates the exploit attempt itself did not need to look successful to the server's own error handling — the SQL injection achieved its effect as a side channel of a request the application otherwise rejected as malformed. Security teams reviewing Metabase access logs for this pattern should treat any match as strong evidence of a completed compromise per Metabase's own guidance, not merely a probe, since a subsequent 200 response from the current-user endpoint indicates a live authenticated session was established.

Recommendations

Immediate Actions

Organizations running any self-hosted Metabase instance between version 58.0 and the pre-patch 63.x builds should upgrade immediately to the minimum safe release for their branch — 58.24, 59.21, 60.17, 61.11, 62.9, or 63.5 — regardless of whether they have observed suspicious activity, since the absence of an assigned CVE means many vulnerability scanners will not surface this exposure on their own [3][8]. Metabase Cloud customers were patched automatically as part of the vendor's response and do not need to take upgrade action, though the post-patch verification steps below still apply if the instance's `reset_password` endpoint was internet-facing before the fix. Teams that cannot patch within the next few hours should block or rate-limit the `/api/session/reset_password` endpoint at a web application firewall, reverse proxy, or network perimeter as an interim compensating control, understanding that this only closes the specific exploited path and does not address any other latent risk in the same code [2][3].

Short-Term Mitigations

Any instance that exposed the `reset_password` endpoint to the public internet prior to patching should be treated as potentially compromised and taken through Metabase's full post-incident checklist rather than a simple version bump. That checklist includes deleting all rows in the `core_session` table to invalidate any sessions an attacker may have established, reviewing and removing any API keys the organization does not recognize as its own, auditing the full list of administrator accounts for unauthorized additions, rotating credentials for every database or warehouse connected to the instance, and inspecting connected-warehouse access logs and Metabase's own query history for evidence of bulk data export [2][3]. Organizations should search their Metabase access logs specifically for the `POST`

`/api/session/reset_password` (400) followed by `GET /api/user/current` (200) pattern described above, and treat any match as a confirmed incident requiring credential rotation and customer notification review rather than a false positive to be dismissed.

Strategic Considerations

This incident is consistent with a pattern this note's authors have observed elsewhere: internally deployed business intelligence and analytics tools are often risk-classified as low-priority internal utilities, even though, by design, they aggregate credentials to an organization's most sensitive data stores in one place. Security teams should inventory every self-hosted BI, reporting, or dashboarding tool with the same rigor applied to production application servers, verify none of them expose administrative or account-recovery endpoints directly to the internet without additional network controls, and confirm that credentials such platforms hold for downstream databases are scoped to least privilege rather than broad administrative access to the warehouse itself. Because this flaw was discovered through active-exploitation telemetry rather than a coordinated disclosure process, and because it carried no CVE at the time attackers were already using it, organizations should also treat vendor security-advisory and GitHub Security Advisory feeds — not just NVD-keyed scanners — as a required input to patch-management programs for self-hosted open-source software.

CSA Resource Alignment

This incident is a direct illustration of the discovery-to-remediation gap that CSA's Project Glasswing: AI Discovery Outpaces Open Source Patching Capacity documents at ecosystem scale. That research found that exploitation timelines have compressed to hours after disclosure while patch development and rollout for open-source projects still take weeks, and it identifies vulnerability remediation — not discovery — as the primary bottleneck in modern application security. The Metabase case exhibits the same pattern in miniature: the vendor detected and patched its own Cloud fleet within roughly three days of active exploitation, but self-hosted operators running any of six affected release branches remained exposed until they applied a fix themselves, and the absence of an assigned CVE at disclosure time means many organizations' patch-tracking tooling will not have flagged the gap automatically.

CSA's 2026 State of Modern Application & AI Security survey, based on responses from more than 900 practitioners, separately found that organizations' application security programs frequently break down specifically in production — where pre-deployment controls no longer apply and runtime exposure depends on how quickly a known or newly disclosed flaw gets patched. The Metabase incident is a case

study in exactly that failure mode: a production-deployed, internet-facing analytics tool with a defect in a rarely-audited authentication-adjacent code path, exploited in the field before the vendor's own security process caught up to it.

Both the injection vector itself and the resulting administrator-level compromise map to the Threat and Vulnerability Management and Application and Interface Security domains of CSA's AI Controls Matrix (AICM) v1.1. Although Metabase's flaw is a conventional SQL injection rather than an AI-specific vulnerability, organizations that have embedded Metabase or similar BI tools into AI-assisted analytics or agentic reporting pipelines should apply the AICM's control objectives for vulnerability identification, timely remediation, and secure interface design to any component — AI-enabled or not — that holds standing credentials to connected data sources, since the practical risk demonstrated here is that a single compromised reporting layer can expose every downstream system it was built to summarize.

References

[1] The Hacker News. "Metabase Zero-Day Exploited in Wild Allows Admin Access Without Authentication." The Hacker News, August 2026.

[2] Security Affairs. "Metabase Zero-Day Exploited in the Wild, Exposing Admin Access and Sensitive Data." Security Affairs, August 2026.

[3] Metabase. "SQL injection using an unauthenticated endpoint leading to admin access (GHSA-vwf4-m7j8-wcjf)." GitHub Security Advisories, August 2026.

[4] Metabase. "Security update available for Metabase - Please upgrade now." Metabase Blog, August 2026.

[5] Cyberpress. "Metabase Zero-Day Attack Lets Hackers Gain Admin Access and Steal Database Credentials." Cyberpress, August 2026.

[6] BleepingComputer. "Metabase SQLi zero-day exploited in customer data-theft attacks." BleepingComputer, August 2026.

[7] Cyber Security News. "Metabase 0-Day Vulnerability Exploited in the Wild to Gain Admin Access." Cyber Security News, August 2026.

[8] Security Online. "Metabase SQL Injection Zero-Day (CVSS 10) Exploited." Security Online, August 2026.

