Metabase Unauthenticated SQL-Injection Zero-Day

CVE-2026-72898 (CVSS 10.0) — A Defender's Deep Report

Status at time of writing (2026-08-12): actively exploited, confirmed against at least five organizations, patched by the vendor, and listed on CISA's Known Exploited Vulnerabilities (KEV) catalog with a federal remediation deadline of August 14, 2026.

This report explains the technical root cause, the affected/fixed version matrix, independently verifiable evidence of in-the-wild exploitation, detection guidance, interim mitigations, and open questions — without exploit code or attack instructions.

Prepared for security engineers, incident responders, and platform/DevOps teams running self-hosted Metabase.

CVE-2026-72898 • CVSS 10.0 • CISA KEV listed • Due 2026-08-14

Contents

Status at time of writing (2026-08-12): Actively exploited, confirmed against at least five organizations, patched by the vendor, and listed on CISA's Known Exploited Vulnerabilities (KEV) catalog with a federal remediation deadline of **2026-08-14**. This report is written for security engineers, incident responders, and platform/DevOps teams who run self-hosted Metabase and need to assess exposure, detect prior compromise, and remediate — without needing exploit code to do any of that.

1. Executive summary

On **August 6, 2026**, Metabase — the widely deployed open-source and Enterprise business-intelligence (BI) platform — disclosed that its own Metabase Cloud infrastructure had been breached by an attacker exploiting a previously unknown ("zero-day") vulnerability in the password-reset API (Metabase security update). The flaw, now tracked as **CVE-2026-72898**, lets a completely unauthenticated, remote attacker inject arbitrary SQL into the Metabase *application* database through the `POST /api/session/reset_password` endpoint, and use that injection to seize full administrator control of the instance (GitHub Security Advisory GHSA-vwf4-m7j8-wcjf).

Both NVD/CVE.org and the GitHub advisory rate the flaw the maximum possible severity, **CVSS v3.1 10.0** (`AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H`), and CISA's own CVE record additionally publishes a CVSS v4.0 score of **10.0** and a Stakeholder-Specific Vulnerability Categorization (SSVC) rating of "Exploitation: active, Automatable: yes, Technical Impact: total" (CVE-2026-72898 record, cveawg.mitre.org). On **August 11, 2026**, CISA added CVE-2026-72898 to its Known Exploited Vulnerabilities catalog with a due date of **August 14, 2026**, confirming independently that active exploitation is occurring in the wild (CISA KEV catalog feed; CISA KEV JSON, cveID CVE-2026-72898).

Because Metabase instances routinely hold live connection credentials for the analytical databases and data warehouses they front — Postgres, MySQL, Snowflake, BigQuery, Redshift, MongoDB, Databricks, and so on — admin takeover of Metabase is a springboard into every downstream data store it touches (Wiz, "Inside the Metabase SQLi"). At least five organizations have publicly confirmed data theft traced to this flaw: **Framework** (laptop manufacturer), **n8n** (workflow-automation platform), **Tally** (form-builder SaaS), **Kilo Code** (AI coding assistant, owned by Anaconda), and **ChecklyHQ** (monitoring/testing platform) (Help Net Security; BleepingComputer). All five incidents trace back to the same root cause: their vendor's or their own Metabase Cloud/self-hosted instance was reachable and unpatched when the attacker's campaign hit between **August 2–3, 2026**.

Metabase shipped fixes for six major-version branches (58 through 63) on **August 6, 2026**, and shipped a second, broader hardening release (GHSA-r495-55cx-fjh7) on **August 11, 2026** that closes a family of related SQL-construction and permission-check gaps — including a dependency update to Metabase's SQL-generation library, HoneySQL, to remediate a separate upstream CVE (GHSA-r495-55cx-fjh7). Two sibling SQL-injection issues disclosed the same day as CVE-2026-72898 — **CVE-2026-72899** (SQLi via public dashboards) and **CVE-2026-72900** (cross-user data leakage) — are not (as of this writing) listed in CISA KEV, meaning only CVE-2026-72898 has independently confirmed in-the-wild exploitation; the other two should still be treated as urgent because they share the same patch and are trivially reachable once an attacker knows the pattern.

This report explains the technical root cause in defender-relevant terms, gives the complete affected/fixed version matrix, lays out exactly what independently verifiable evidence exists for in-the-wild exploitation and its scope, provides concrete detection guidance (log signatures and indicators), lists interim mitigations for teams that cannot patch immediately, and closes with the open questions that remain unresolved at time of writing. **It intentionally omits exploit code, working SQL payloads, or step-by-step attack instructions.**

2. What actually happened — the incident timeline

The timeline below is reconstructed from Metabase's own advisory, the affected companies' individual breach notifications, and CISA's KEV entry. Where a date is attributed to a specific company, that company's own disclosure is cited.

Date (2026)	Event	Source
Aug 2, ~4 hours	Attacker exploits the zero-day against Metabase Cloud infrastructure itself, and separately (per victim disclosures) against tenant instances	Metabase security update; Wiz
Aug 3	Attacker accesses Framework's, Tally's, and ChecklyHQ's Metabase-connected data; Checkly says the session lasted roughly 26 minutes and was read-only	Framework notice, via TechCrunch; Checkly, "Metabase Security Incident"
Aug 6	Metabase identifies the 0-day, blocks the abused endpoints, patches all six affected major-version branches (58–63), and begins notifying affected customers (Framework notified 9am Pacific)	Metabase security update; TechCrunch
Aug 6	GitHub Security Advisories GHSA-vwf4-m7j8-wcjf (CVE-2026-72898), GHSA-r8h2-qpfx-mx59 (CVE-2026-72899), and GHSA-8hmm-hrhg-ppqp (CVE-2026-72900) published	GitHub Advisory Database
Aug 7–8	Framework and Tally publish customer breach notices; n8n publishes initial disclosure	BleepingComputer; n8n blog
Aug 10	Anaconda (Kilo Code) and ChecklyHQ publish breach notices; Wiz publishes technical root-cause writeup; public proof-of-concept exploit code appears	Anaconda blog; Checkly blog; Wiz
Aug 10	CVE-2026-72898 formally published by CISA (acting as CNA) via CVE.org	MITRE CVE record
Aug 11	CISA adds CVE-2026-72898 to the KEV catalog; due date set to Aug 14, 2026	CISA KEV catalog

Date (2026)	Event	Source
Aug 11	n8n publishes forensic update: 136 customer records confirmed accessed	n8n blog update
Aug 11	Metabase ships a second, broader hardening release (GHSA-r495-55cx-fjh7) covering ten categories of related fixes, including a HoneySQL dependency bump	GHSA-r495-55cx-fjh7

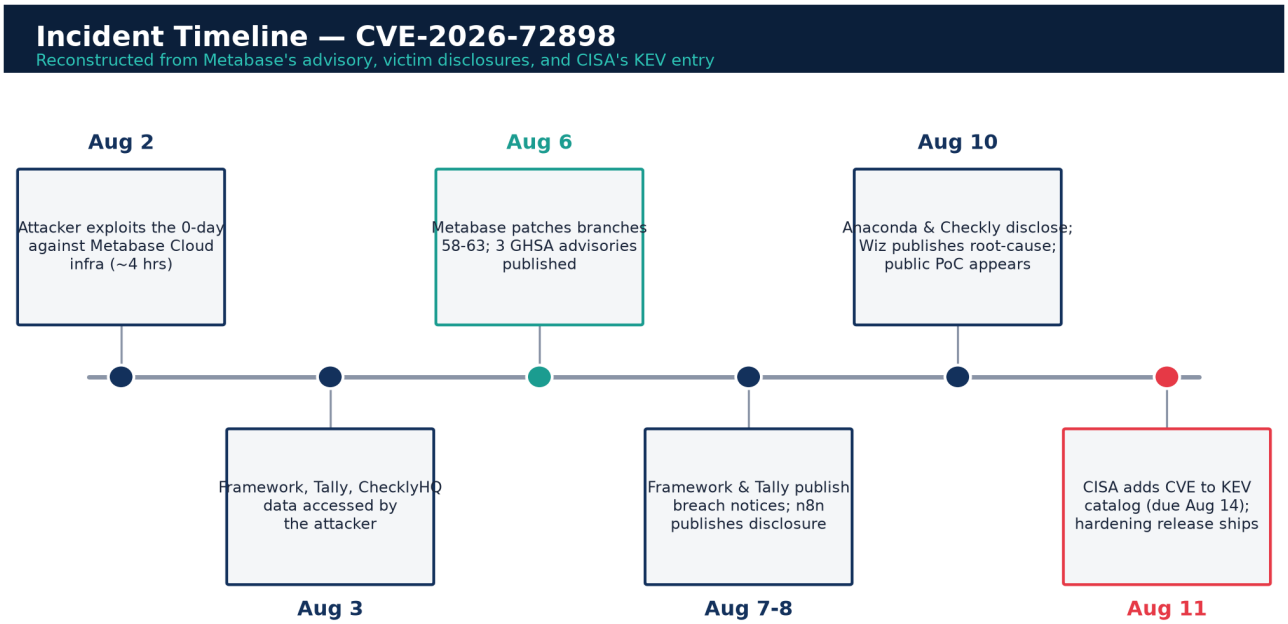

Figure 1. Incident timeline reconstructed from the vendor advisory, victim disclosures, and CISA's KEV entry.

A note on precision: Metabase's own post says the attack against its Cloud infrastructure "occurred over approximately 4 hours" on August 2, and several tenants report follow-on access on August 3 — consistent with an attacker who first found the bug against Metabase's own perimeter and then pivoted to (or separately scanned for and hit) individual customer-hosted or Cloud-hosted instances. No source has published a single authoritative minute-by-minute timeline spanning all five victims; the table above is the best independently corroborated reconstruction available.

3. The underlying weakness: how the SQL injection actually works, and why it occurred

3.1 The vulnerable surface

The primary flaw (CVE-2026-72898) lives in the unauthenticated password-reset endpoint, `POST /api/session/reset_password` — by design, an endpoint that must be reachable *without* a session, because its entire purpose is to let a logged-out user reset their password with a token. NVD's classification is **CWE-89, Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')** (NVD summary via MITRE CVE record), and CISA's own KEV description states plainly: *"Metabase contains a SQL Injection vulnerability that allows an unauthenticated remote attacker to inject arbitrary SQL into the Metabase application database, which can give them administrator access to the instance."* (CISA KEV catalog JSON feed).

3.2 Root cause, in the vendor's own words

Bishop Fox's technical writeup states the mechanism concisely: *"The root cause is a failure to restrict undeclared fields in the password-reset request body,"* which lets an attacker-supplied value reach an application-database user lookup as structured input instead of a validated identifier (Bishop Fox, "Critical SQL Injection in Metabase via Password Reset").

Wiz's independent technical analysis fills in the mechanism at the code level. Metabase is written in Clojure and builds its SQL with the HoneySQL library. Per Wiz's writeup:

- The reset-password handler accepts a JSON body and internally **merges** the request data with the result of an authentication/token-lookup step, without stripping keys that shouldn't be there. Clojure's `merge` function does not restrict which keys survive the merge.
- The endpoint's JSON parser converts request keys to Clojure keywords ("keywordization"). A JSON body containing an undocumented `user-id` key whose value is itself a nested object — rather than the plain integer the code expects — survives intact through the merge when the token/authentication step fails, and is carried forward as a Clojure map.
- HoneySQL, the SQL-generation library, has a documented feature: a map of the shape `{:raw "<string>"}` is a legitimate escape hatch developers use to drop a literal, unparameterized SQL fragment into a generated query, deliberately bypassing HoneySQL's normal parameter-binding/escaping. When the attacker-controlled value produced by the merge above reaches a HoneySQL query as `{:raw "<attacker string>"}`, HoneySQL faithfully honors it as a directive to splice the string into the SQL verbatim — with no quoting and no bound parameter.

(Source: Wiz, "Inside the Metabase SQLi: Exploited in the Wild".)

In other words, this is not a "forgot to escape a string" bug in the traditional sense — it is a **type-confusion-enabled injection**: an internal library feature intended for trusted, developer-authored SQL fragments (HoneySQL's `:raw`) became reachable from attacker-controlled, unauthenticated HTTP input because an intermediate data-merging step failed to validate that a field which was *supposed* to be a plain integer actually was one before it flowed into the query-building layer. Metabase's own fix, per Wiz, is to validate that `user-id` is a positive integer before it is used in any query — closing the type-confusion gap rather than attempting to sanitize arbitrary structured input after the fact.

The GitHub advisory's own remediation language corroborates the "undeclared field" framing directly, instructing administrators to verify post-patch *"that undeclared fields sent to `/api/session/reset_password` are now rejected"* (GHSA-vwf4-m7j8-wcjf).

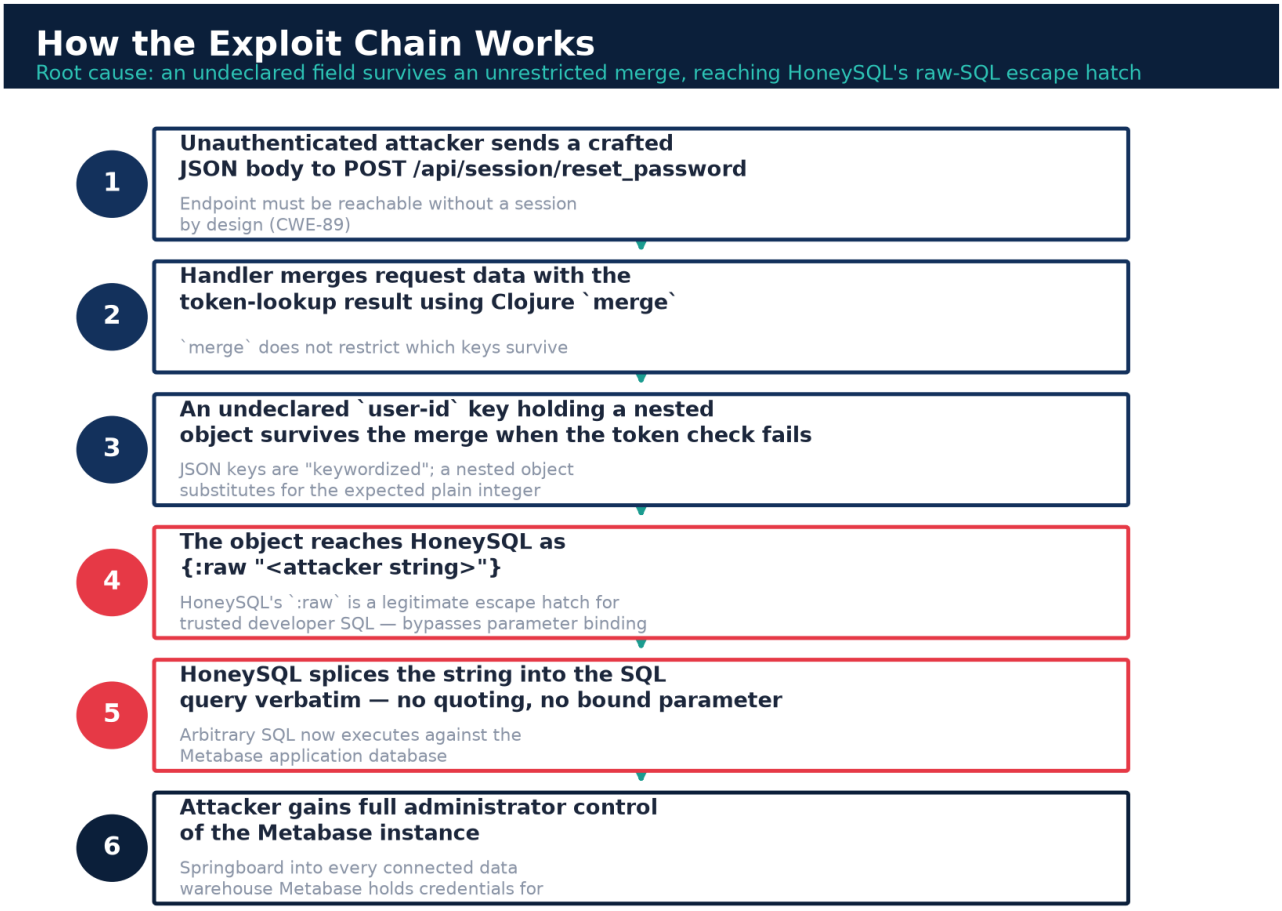

Figure 2. The six-step exploit chain from unauthenticated request to admin takeover.

3.3 Why this class of bug kept recurring in Metabase during 2026

This is not an isolated incident. The same GitHub Security Advisory feed shows a dense cluster of critical- and high-severity Metabase vulnerabilities disclosed across 2026, several involving the same general theme — attacker-influenced values reaching the SQL-compilation layer, or the H2/JDBC driver layer, without adequate boundary checks:

GHSA ID	Published	Severity	Summary
GHSA-r495-55cx-fjh7	2026-08-11	Critical	Multiple vulnerabilities across query validation, permissions, and network exposure (proactive hardening release)
GHSA-r8h2-qpfx-mx59 (CVE-2026-72899)	2026-08-06	Critical	SQL injection using a publicly shared dashboard leading to admin access
GHSA-8hmm-hrhg-ppqp (CVE-2026-72900)	2026-08-06	Medium	Leaking sensitive data from the application database to low-privilege Metabase users

GHSA ID	Published	Severity	Summary
GHSA-vwf4-m7j8-wcjf (CVE-2026-72898)	2026-08-06	Critical	SQL injection using an unauthenticated endpoint leading to admin access — **the flaw covered in this report**
GHSA-cwxq-fmxq-jv8h	2026-07-12	Critical	Arbitrary file read/write via unsafe H2 built-in functions
GHSA-w95f-x9v9-wv36 (CVE-2026-59827)	2026-06-30	Critical	Unsafe deserialization of H2 query results
GHSA-8wx2-rxp2-4x35 (CVE-2026-59826)	2026-06-30	Critical	Arbitrary code execution via database connection detail bypass
GHSA-r6x2-rchx-q9g9 (CVE-2026-50148)	2026-05-28	Critical	Remote code execution via Snowflake JDBC driver arbitrary file write
GHSA-mfpj-crjq-xrcp (CVE-2026-50147)	2026-05-28	High	Arbitrary file read via MySQL connection property injection
GHSA-fppj-vcm3-w229 (CVE-2026-33725)	2026-03-24	High	RCE and arbitrary file read via H2 JDBC INIT injection in EE serialization import

(Full list independently retrieved from GitHub's public Security Advisories API for the metabase/metabase repository.)

Notably, Metabase's own August 11 hardening advisory explicitly frames itself as closing "the remaining places where a value carried by a saved question could reach the database as raw text," across query parameters/filters, cast options, stored metric/segment definitions, custom column types, and warehouse-specific identifier quoting/escaping for ClickHouse, BigQuery, and Postgres — and separately notes it *"updated the SQL-generation library Metabase depends on (HoneySQL) to a release that patches a recently published vulnerability in that library (CVE-2026-61620)"* (GHSA-r495-55cx-fjh7). This is a direct admission that the underlying pattern — user-influenced values reaching a dynamic, multi-warehouse SQL-compilation layer that must translate one query language into six or more different backend dialects — is architecturally hard to fully close off, and that the August incident forced a systematic audit of that entire layer rather than a single point fix.

Why this matters for defenders: a BI tool that (a) must accept flexible, structured user input to build dashboards and filters, (b) must translate that input into native SQL for a dozen different database engines, and (c) stores live credentials to every one of those engines, is an unusually attractive and unusually fragile target. The same architecture that makes Metabase powerful (a generic query-compilation layer spanning Postgres, MySQL, Snowflake, BigQuery, Redshift, ClickHouse, MongoDB, Databricks, SQL Server, and H2) is exactly what makes each new input-validation gap in that layer a full-severity vulnerability rather than a narrow one.

4. Full affected and fixed version matrix

Metabase ships two parallel product lines with matching version numbers: the open-source (OSS) edition uses a `0.x` prefix and the Enterprise Edition (EE) uses a `1.x` prefix for the same underlying version number (e.g., OSS `0.58.23` and EE `1.58.23` share the same codebase and vulnerability status). The table below consolidates the three CVEs disclosed on August 6, 2026, all of which affect the same six major-version branches and are fixed by the same six point releases.

Major branch	Vulnerable range (OSS / EE)	Fixed version	Applies to
58	0.58.0–0.58.23 / 1.58.0–1.58.23	0.58.24 / 1.58.24	CVE-2026-72898, -72899, -72900
59	0.59.0–0.59.20 / 1.59.0–1.59.20	0.59.21 / 1.59.21	CVE-2026-72898, -72899, -72900
60	0.60.0–0.60.16 / 1.60.0–1.60.16	0.60.17 / 1.60.17	CVE-2026-72898, -72899, -72900
61	0.61.0–0.61.10 / 1.61.0–1.61.10	0.61.11 / 1.61.11	CVE-2026-72898, -72899, -72900
62	0.62.0–0.62.8 / 1.62.0–1.62.8	0.62.9 / 1.62.9	CVE-2026-72898, -72899, -72900
63	0.63.0–0.63.4 / 1.63.0–1.63.4	0.63.5 / 1.63.5	CVE-2026-72898, -72899, -72900

Sources for this matrix (cross-checked and consistent across all three): MITRE CVE record for CVE-2026-72898 (machine-readable `affected.versions` block), GHSA-vwf4-m7j8-wcjf, GHSA-r8h2-qpfx-mx59, GHSA-8hmm-hrhg-ppqp, and Metabase's own security-update post, which states in plain language: *"Versions 1.58 and above are vulnerable. Versions below 58 are unaffected."*

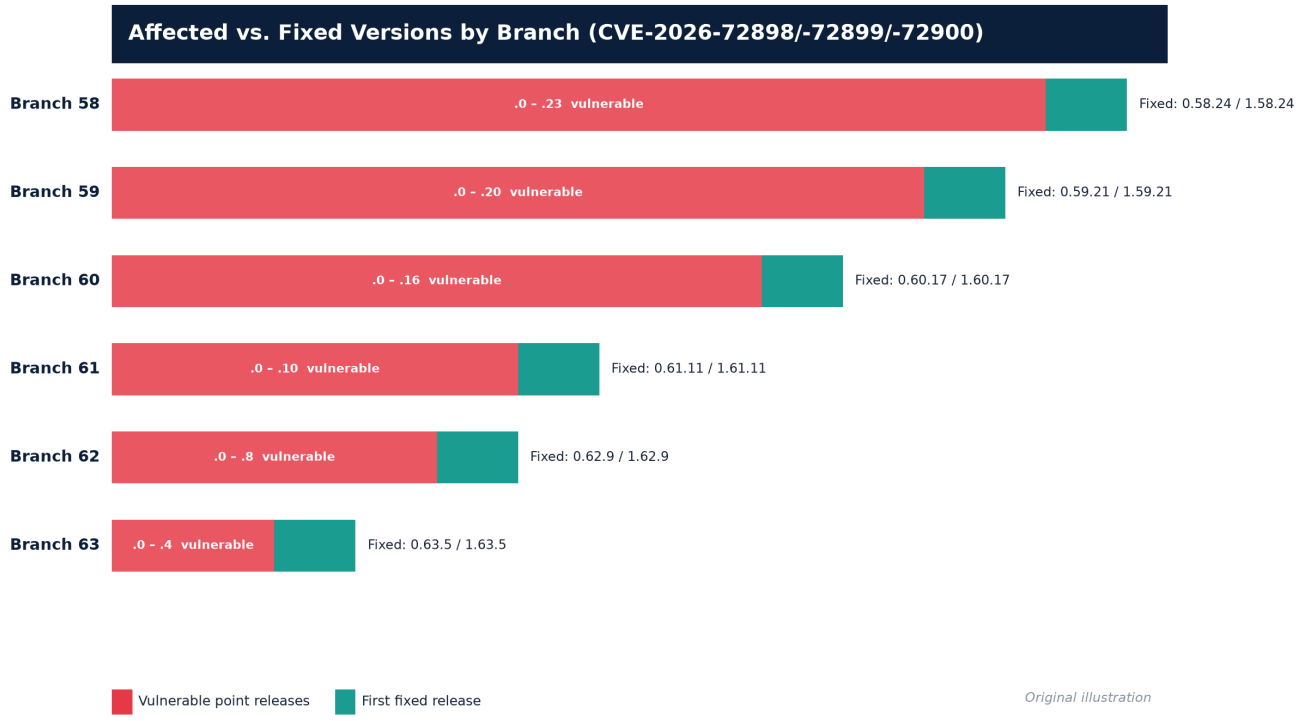

Figure 3. Vulnerable point-release ranges versus first-fixed release, per major branch.

Important scope notes:

- **Versions below branch 58 (i.e., 57.x and earlier) are not affected** by any of the three August 6 CVEs, per Metabase's own statement. (Older branches have their own, unrelated CVE history — see §3.3 for context on the January–March 2026 advisories affecting 57.x-era code, none of which are part of this incident.)
- **Metabase Cloud customers do not need to take action for the version-upgrade step.** Metabase's advisory states unambiguously: *"If you are a Metabase Cloud customer, your instance is already upgraded and patched"* (Metabase security update). However, Cloud customers whose instances were *compromised before* the patch (i.e., any of the five publicly confirmed victims, or others not yet disclosed) still need to complete the post-compromise response steps in §7 below — patching does not undo data already exfiltrated.
- **Self-hosted OSS and EE customers on any branch 58–63 below the fixed point release remain fully vulnerable** until upgraded, and this is the population CISA's KEV listing and BOD 26-04 obligations are aimed at.
- A second, non-emergency hardening release, **GHSA-r495-55cx-fjh7**, was published August 11, 2026, bumping the *fixed* target versions further, to **0.58.28 / 1.58.28, 0.59.25 / 1.59.25, 0.60.21 / 1.60.21, 0.61.15 / 1.61.15, 0.62.13 / 1.62.13, and 0.63.10 / 1.63.10**. This release has no CVE identifier of its own (it is framed by Metabase as proactive hardening, not a disclosed incident), but defenders patching now should go straight to these later point releases rather than stopping at the August 6 minimums, since the August 11 release also patches the upstream HoneySQL library vulnerability (CVE-2026-61620) that the same code path depends on (GHSA-r495-55cx-fjh7).

5. Severity scoring, in detail

Metric	CVE-2026-72898 (password reset)	CVE-2026-72899 (public dashboard)	CVE-2026-72900 (cross-user leak)
CVSS v3.1 vector	`AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H`	`AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H`	`AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`
CVSS v3.1 score	**10.0 Critical**	**9.6 Critical**	**6.5 Medium**
CVSS v4.0 score (CISA)	**10.0 Critical**	not published by CISA	not published by CISA
Authentication required	None	None (needs only a public link UUID, which is exposed by design once a dashboard/card is shared)	Low-privileged authenticated account
In CISA KEV as of Aug 12, 2026	**Yes** — dateAdded 2026-08-11	No	No

Sources: MITRE/CISA CVE record for CVE-2026-72898; GHSA-vwf4-m7j8-wcjf; GHSA-r8h2-qpfx-mx59; GHSA-8hmm-hrhg-ppqp; CISA KEV JSON feed, filtered to CVE-2026-72898.

CVE-2026-72900's own advisory text is worth quoting because it explains why the *medium*-severity number understates its practical danger when chained with the other two flaws: *"On any instance without* `MB_ENCRYPTION_SECRET_KEY` *set (the default), the same read returns the stored credentials for every connected database in cleartext"* (GHSA-8hmm-hrhg-ppqp). Metabase does not enable application-database field encryption out of the box; an operator who never set that environment variable is one authenticated low-privilege session away from a cleartext credential dump even without CVE-2026-72898.

CWE classification: CVE-2026-72898 is classified CWE-89 (SQL Injection) by both the CVE Numbering Authority record and CISA's KEV entry (CISA KEV JSON feed).

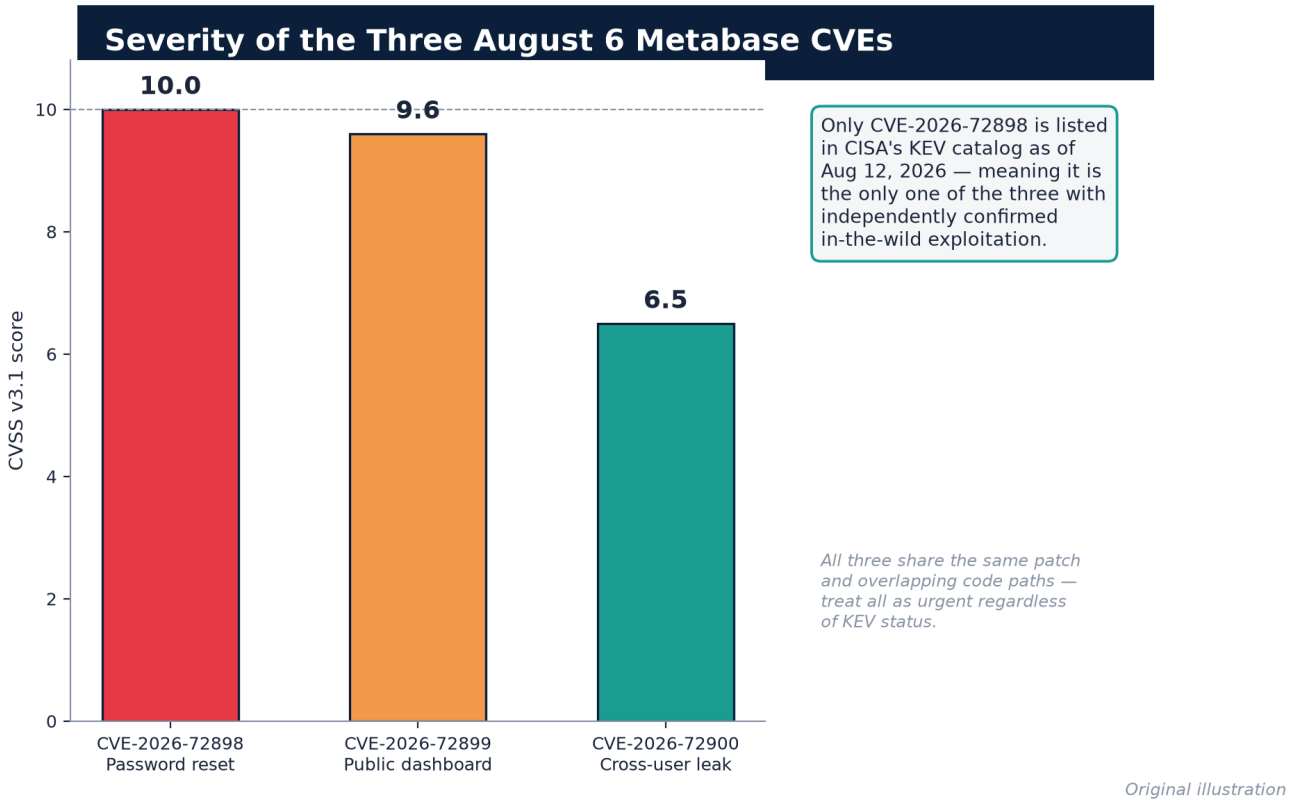

Figure 4. CVSS v3.1 severity comparison across the three August 6, 2026 Metabase CVEs.

6. Is it being exploited in the wild, and how widely? What defenders can independently verify

This section separates **confirmed, independently verifiable facts** from **claims that are plausible but not independently corroborated**, because that distinction matters for how much weight a defender should put on each data point.

6.1 Confirmed by an authoritative third party: CISA KEV

CISA does not add a CVE to the KEV catalog on vendor say-so alone; KEV listing requires CISA to have independent or corroborated evidence of active exploitation. The KEV entry for CVE-2026-72898, retrieved directly from CISA's machine-readable feed, states:

> *"Metabase contains a SQL Injection vulnerability that allows an unauthenticated remote attacker to inject arbitrary SQL into the Metabase application database... Apply mitigations in accordance with vendor instructions, ensuring compliance with CISA's BOD 26-04... Follow applicable BOD 26-04 guidance for cloud services or discontinue use of the product if mitigations are unavailable."*

Field `knownRansomwareCampaignUse` is currently set to `"Unknown"` — meaning CISA has *not* attributed this vulnerability to a tracked ransomware operation as of this writing, only to data-theft activity (CISA KEV catalog JSON feed). This is a meaningful, independently checkable fact: defenders should not currently frame this as a ransomware precursor absent further evidence, only as a confirmed data-exfiltration vector.

6.2 Confirmed by named victims (five organizations, independently disclosed)

Each of the following disclosures is a first-party statement from the affected company, independently published and independently verifiable by visiting the source:

- **Framework** (laptop manufacturer): names, email addresses, phone numbers, physical addresses, and login IP addresses accessed; for business customers, potentially company names, VAT/EIN numbers, and billing emails. Framework states payment and order details were *not* affected. (Framework disclosure, via TechCrunch; The Register)
- **n8n** (workflow automation): 136 customer records accessed across self-hosted and Cloud users; 5 of those records included bcrypt-hashed n8n Cloud passwords; n8n separately disclosed an unrelated historical bug that had stored ~25 Cloud account passwords in plaintext, discovered during this incident's forensic review. n8n notified the Berlin Commissioner for Data Protection and Freedom of Information. (n8n blog, "Metabase security incident update")
- **Tally** (form-builder SaaS): customer email addresses and password hashes accessed; Tally states form content and submitted responses were *not* affected because they live in a separate database. (BleepingComputer)
- **Kilo Code / Anaconda** (AI coding assistant): a subset of Kilo Slackbot users had Slack access tokens exposed; a subset of users/organizations may have had prompts or user data (name, email, billing address, location) exposed. Anaconda invalidated all Kilo Slackbot tokens for affected users as a precaution. (Anaconda blog, "Metabase Incident Impacting Kilo Code Data")
- **ChecklyHQ** (monitoring/testing platform): attacker obtained a read-only administrator session and queried Checkly's connected data warehouse for approximately 26 minutes on August 3; Checkly states no password was stolen and its production platform was not directly breached, but recommends rotating credentials that were stored in check configurations or used as OTEL API keys. (Checkly blog, "Metabase Security Incident")

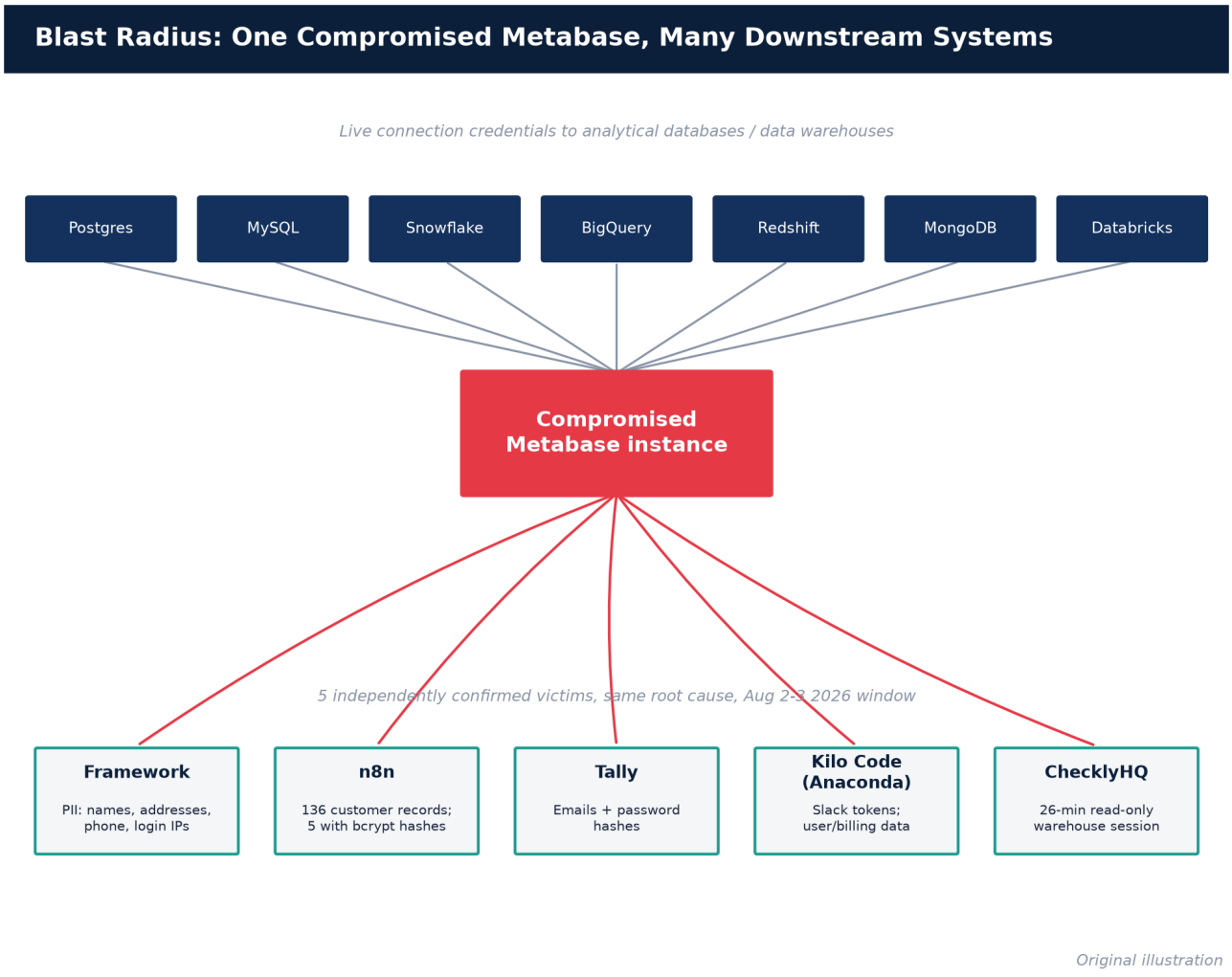

Figure 5. Blast radius: one compromised Metabase instance, its downstream database credentials, and the five confirmed victims.

Five *independently operated* companies confirming compromise via the same disclosed technique, within the same 48-hour window, is itself strong corroborating evidence of a real, broad exploitation campaign — this is not a single company's claim.

6.3 Exposure scale (how many instances are theoretically reachable) — reported by Wiz, not independently re-verified here

Cloud security vendor Wiz published exposure statistics in its technical writeup: approximately **13% of surveyed cloud environments have a self-hosted Metabase instance deployed**, of which roughly **25% are fully internet-accessible**, and **Shodan inventories approximately 2,500 Metabase instances** visible on the public internet (Wiz, "Inside the Metabase SQLi"). These figures were also independently repeated in vendor/press coverage (e.g., CSO Online). Note the important caveat: these are Wiz's own telemetry and Shodan's index, not a figure independently reproduced by a second, unrelated measurement source in the material reviewed for this report — defenders relying on this number should treat it as a **directionally credible but single-source estimate**, not an audited count, and should re-run their own asset inventory

(see §8) rather than relying on it for their own exposure decision.

6.4 What is *not* independently confirmed as of this writing

- **No public report from GreyNoise or Shadowserver** specifically quantifying internet-wide scanning/exploitation *volume* for CVE-2026-72898 was found among the sources reviewed for this report as of August 12, 2026. This does not mean no such scanning is occurring — GreyNoise and Shadowserver often publish such telemetry with a short lag after KEV listing — but as of this writing, defenders do not yet have an independently measured "how many attackers are actively scanning the internet for this" figure to cite; only the five confirmed-victim disclosures and Wiz's exposure/Shodan counts.
- **Public proof-of-concept exploit code is confirmed to exist** (Wiz states it observed public PoCs "as of noon UTC on August 10th"), which materially raises the risk of copycat/opportunistic exploitation beyond the original attacker, but the volume of such follow-on activity has not been independently quantified in a published report at time of writing (Wiz).
- **Attribution is unknown.** No source reviewed identifies who is behind the original campaign, their motive beyond data theft, or whether the five confirmed victims were specifically targeted or opportunistically found via internet-wide scanning. Treat any claim of attribution you see elsewhere with skepticism until CISA, the vendor, or a named incident-response firm publishes one.

7. Detection guidance

7.1 The log signature Metabase itself published

This is the single most concrete, vendor-confirmed detection signature available, and it should be the first thing every self-hosted operator checks. Metabase's own advisory states:

> *"If you find [a]* `POST /api/session/reset_password` *[request] with a 400 status code followed by [a] call to* `GET /api/user/current` *with a 200 status code[,] in your application logs or in your Metabase server ingress logs, it is likely that your instance has been compromised."*

(Source: Metabase security update, corroborated independently by The Hacker News, Security Affairs, and IONIX.)

Why this pattern is meaningful: a legitimate password-reset attempt with a bad or expired token also returns HTTP 400, so the 400 alone is not diagnostic. What is diagnostic is the *pairing*: a 400 on the reset-password call immediately followed by a **successful (200)** call to `/api/user/current` — which is the endpoint an authenticated session uses to fetch "who am I." A genuine failed password reset should never be immediately followed by a successful authenticated session lookup; that sequence is the fingerprint of the attacker's injected session/token being accepted despite the outer reset call reporting failure.

Where to look: reverse-proxy/ingress logs (nginx, ALB/ELB, Cloudflare, whatever fronts your Metabase deployment), and Metabase's own internal request logs if centrally shipped. Because the attack requires no authentication, it will appear in access logs before any Metabase-side audit-log entry exists (the attacker has no account yet at the moment of the injection).

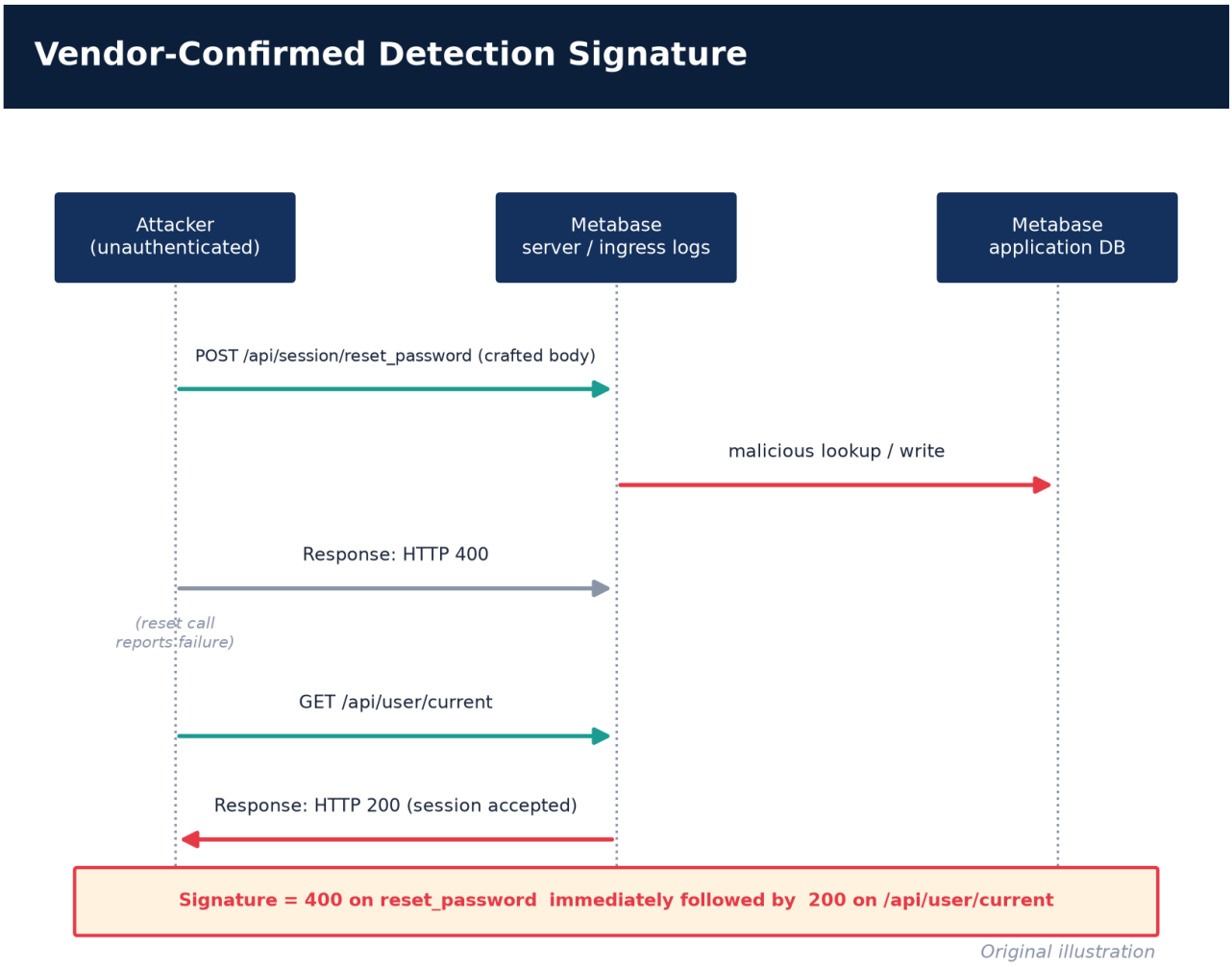

Figure 6. The vendor-confirmed HTTP 400→200 log-pairing detection signature.

7.2 Complementary indicators recommended by third-party analyses

Independent detection guides (IONIX, Security Arsenal, runZero) converge on the same core signature and add secondary indicators worth hunting for, though these are vendor/researcher recommendations rather than confirmed IOCs from an actual observed intrusion, so treat them as *hunting hypotheses*, not confirmed attacker TTPs:

- **Unexpected administrator accounts** created, or existing admin accounts with unexpected email-address changes, password resets, or permission changes — check this directly against Metabase's own `core_user` table / admin panel, not just logs (Bishop Fox).

- **Unrecognized API keys** in the Metabase admin settings — the vendor's own remediation checklist explicitly calls this out as a review step in all three of the August 6 advisories (GHSA-vwf4-m7j8-wcjf).
- **Anomalous query/activity history** inside Metabase itself — Metabase retains a query-history and activity log; review it for queries against system/audit tables, or against databases the account normally never touches, in the days around your suspected exposure window.
- **Downstream data-warehouse/database logs** — because the entire point of this vulnerability is to pivot from Metabase into connected databases, your warehouse-side query logs (Snowflake query history, Postgres `pg_stat_statements`/audit logs, BigQuery audit logs, etc.) for the **service account Metabase uses to connect** are an independent, harder-to-tamper-with source of truth. Look for connections from Metabase's service account executing queries it would not normally run, at times correlated with the August 2–6 window, or outside your organization's normal reporting schedule.
- **Bulk export volume** from `/api/download/` or `/api/dataset/` endpoints — a spike in export volume from a single source, especially outside business hours or from an unfamiliar IP, is a coarse but useful signal (Security Arsenal detection guide).
- **Version fingerprinting without exploitation** — you can identify a Metabase instance's version passively, by requesting `/api/session/properties` (an unauthenticated, informational endpoint) and checking the returned `version` field against the matrix in §4, or by looking for the `window.MetabaseBootstrap` object with an embedded version tag in the HTML response, which IONIX and Wiz both use for asset discovery (IONIX; Wiz). This is a legitimate, non-exploitative reconnaissance technique your own team can use to inventory your estate.

7.3 Asset discovery for teams that don't know where all their Metabase instances are

Attack-surface-management vendor runZero published a specific saved-search query for its own platform: `vendor:=Metabase AND product:=Metabase`, to be run against an organization's existing asset inventory (runZero, "Metabase vulnerability CVE-2026-72898: Find impacted assets"). If you use a different CMDB/ASM tool, the equivalent is to search your asset inventory or network-scan data for any host serving the Metabase web UI (default port **3000** for the built-in Jetty server, though many deployments sit behind a reverse proxy on 443) and then version-fingerprint each hit per §7.2 above.

7.4 No published Sigma/YARA/Suricata rule identified

At the time of writing, no source reviewed for this report links to a formally published, community-vetted Sigma rule, YARA signature, or Suricata/network-IDS rule specific to CVE-2026-72898. The detection guidance above (the vendor's own 400→200 log pairing, plus the secondary indicators) is the most concrete, currently available detection method. If your organization uses a SIEM, the highest-value immediate action is to encode the §7.1 log-pairing logic as a correlation rule against your own ingress/reverse-proxy logs, since that is the one signature with direct vendor confirmation.

8. Interim mitigations and hardening for teams that cannot patch immediately

The vendor's own advisories and independent guides converge on a consistent set of steps. These are organized by urgency.

8.1 Immediate (do this today, before patching)

- **Block or restrict the vulnerable endpoint at the network/reverse-proxy layer.** Metabase's own advisory states this explicitly as the sanctioned temporary workaround: *"If you are unable to upgrade ASAP, as a temporary workaround, block the `/api/session/reset_password` endpoint"* (GHSA-vwf4-m7j8-wcjf). Practically, this means adding a reverse-proxy rule (nginx `location` block, ALB listener rule, WAF rule, Cloudflare Worker/rule, etc.) that returns 403/404 for that specific path while leaving the rest of the application reachable — recognizing that this also disables legitimate self-service password resets until you patch.
- **For the sibling public-sharing flaw (CVE-2026-72899), disable public sharing, or unpublish any public links that expose a field-filter/dimension parameter**, per that advisory's own stated workaround (GHSA-r8h2-qpfx-mx59). If you don't use public dashboards/cards at all, disabling the feature entirely in Admin settings removes this attack surface with no functional cost.
- **Take the instance off the open internet if at all possible.** Every independent guide converges on this: place Metabase behind a VPN, ZTNA, or at minimum an authenticated reverse proxy / IP allowlist, rather than leaving the admin UI and API directly internet-facing (Security Arsenal). Given that CISA's own required action explicitly invokes BOD 26-04's cloud-service guidance and states organizations should *"discontinue use of the product if mitigations are unavailable"* for the highest-risk exposure profile, taking the instance offline entirely is a legitimate option to consider if you cannot patch and cannot restrict access.

8.2 Patch, then complete the vendor's full post-upgrade checklist

Patching alone does **not** remediate a prior compromise — it only closes the door going forward. Metabase's advisory is explicit that self-hosted customers should treat any instance that had the vulnerable endpoint reachable as potentially compromised and complete this sequence after upgrading to the fixed version:

- **Revoke all active sessions** — by connecting to the Metabase application database directly and clearing the `core_session` table (Metabase's advisory literally instructs: *"revoke all active user sessions by accessing the Metabase Application Database and deleting all rows in the `core_session` table"*) (Metabase security update).
- **Review and delete unrecognized API keys.**
- **Audit administrator accounts** for any unexpected creation, email change, or permission escalation.
- **Rotate credentials for every connected database and data warehouse** — this is the step that matters most given the whole point of the exploit chain is credential theft. Rotate database passwords/service-account credentials, LDAP bind credentials, SMTP credentials, and any API keys or cloud-warehouse tokens Metabase was configured with.

- **Set** `MB_ENCRYPTION_SECRET_KEY` if it is not already set, closing the cleartext-credential exposure that compounds CVE-2026-72900 (§5 above) (GHSA-8hmm-hrhg-ppqp).
- **Review downstream data-warehouse and database logs** for signs of unauthorized access from Metabase's service account.
- **Review Metabase's own activity and query history** for unexpected or unauthorized activity.

(This sequence is drawn near-verbatim from the remediation sections of all three August 6 GitHub advisories: GHSA-vwf4-m7j8-wcjf, GHSA-r8h2-qpfx-mx59, GHSA-8hmm-hrhg-ppqp.)

8.3 Longer-term hardening (beyond this specific incident)

- **Prefer the August 11 hardening release targets, not the August 6 minimums.** As noted in §4, patch to 58.28/59.25/60.21/61.15/62.13/63.10 or later where feasible, since that release also remediates the upstream HoneySQL CVE-2026-61620 and a broad set of related permission and query-validation gaps (GHSA-r495-55cx-fjh7).
- **Move off the embedded H2 application database to a dedicated Postgres/MySQL backend** for production deployments — this is a long-standing Metabase best practice independent of this incident, and reduces the blast radius of application-database-level bugs (recommended by Security Arsenal and consistent with Metabase's own general production-deployment documentation).
- **Run Metabase's connections to downstream databases with least-privilege, read-only, schema-scoped service accounts** rather than broad admin credentials, so that even a fully successful Metabase compromise limits what an attacker can reach in your actual data warehouse.
- **Centralize Metabase's audit/activity logging to an external SIEM** independent of the Metabase application database itself, so that an attacker with admin access to Metabase cannot also erase the evidence of their own access.
- **Enforce SSO and disable local password authentication** where your Metabase license tier supports it, which removes the password-reset flow (and therefore this entire attack surface) as a login path altogether.

9. Practical meaning for defenders: who should care and how much

- **If you run any self-hosted Metabase OSS or Enterprise instance on branch 58–63 below the fixed version, and it is (or ever was) reachable without a VPN/allowlist**, you are in the highest-priority bucket: assume potential compromise, patch immediately, and run the full post-upgrade checklist in §8.2 rather than just the patch.
- **If you are a Metabase Cloud customer**, your instance is already patched, but you should still check whether *you* are one of the (potentially more than five, undisclosed) tenants who were actually hit during the August 2–6 window — Metabase's disclosure to Framework and others came via direct notification, so if you have not heard from Metabase directly, the company's public statements do not tell you

definitively whether your specific tenant was touched. Consider proactively asking your account team for confirmation.

- **If you are a downstream customer of any of the five confirmed-affected companies** (or any other company that uses Metabase and has not yet disclosed), treat any unexpected password-reset prompts, unusual login notifications, or phishing attempts referencing that vendor with elevated suspicion — Framework explicitly warned its own customers to watch for impersonation phishing following its breach notice (Help Net Security).
- **If you evaluate or procure BI/analytics tooling**, this incident is a concrete illustration of a structural risk in the category: any tool that must hold live credentials to your production data warehouses is a Tier-0 asset from a blast-radius perspective, regardless of how "just for dashboards" its intended use feels. Segregate it accordingly — network isolation, least-privilege downstream credentials, and independent audit logging are not optional hardening for this class of tool.
- **Federal civilian agencies** are bound by the KEV due date of **August 14, 2026** and, per the KEV entry's required action, must additionally follow **BOD 26-04**'s risk-based prioritization and forensic-triage guidance, given this vulnerability meets the directive's high-risk criteria (confirmed active exploitation, no authentication required, automatable, and total technical impact) (CISA KEV JSON feed; BOD 26-04).

10. Open questions

Being explicit about what is *not* yet known is as important as what is known, for a defender deciding how much residual risk to carry:

- **Full scope of victims.** Only five organizations have publicly disclosed compromise as of this writing. Given Wiz's estimate of roughly 2,500 internet-exposed instances, and that public PoC code has existed since at least August 10, it is very likely more organizations were or will be affected but have not yet disclosed (or do not yet know).
- **Attacker identity and motive.** No source reviewed attributes the original campaign to a named threat actor or group. Whether the initial attacker was financially motivated, a researcher who crossed an ethical line, or a state-linked actor is unknown.
- **Whether exploitation was targeted or opportunistic.** It is unclear whether the five known victims were specifically selected, or discovered via broad internet-wide scanning of Metabase's default port/banner. The presence of very different-sized, differently-profiled targets (a laptop hardware company, a workflow-automation SaaS, a form builder, an AI coding tool, a monitoring platform) is at least consistent with opportunistic, scan-driven targeting, but no source confirms this directly.
- **Independent scan/exploitation telemetry.** As of this writing, no GreyNoise or Shadowserver report specific to this CVE was found in the sources reviewed. Defenders should watch for one; its absence now is a gap in the evidence base, not evidence of an absence of scanning.
- **Whether the August 11 "hardening" release (GHSA-r495-55cx-fjh7) fixes were themselves independently, actively exploited**, or were purely proactive. Metabase's own framing ("we proactively

identified and fixed several issues and bugs") suggests the latter, but the company's framing of the original zero-day before disclosure was also initially private, so an independent confirmation either way has not been published.

- **Whether CVE-2026-72899 (public-dashboard SQLi) or CVE-2026-72900 (cross-user leak) have also been exploited in the wild.** Neither is currently listed in CISA KEV, meaning CISA has not (yet) obtained independent evidence of their exploitation — but given they were patched simultaneously with the confirmed-exploited flaw and share considerable code-path overlap, defenders should not assume they are lower-risk in practice, only that the *evidence* of exploitation is currently limited to CVE-2026-72898.
- **Ransomware linkage.** CISA's KEV entry currently lists `knownRansomwareCampaignUse: "Unknown"`. This field is reviewed and updated by CISA over time; it is worth periodically re-checking the live KEV feed for this CVE.

Sources

#	Claim / Fact	Evidence type	Source	URL
1	Official CVE record: description, CVSS v3.1/v4.0, CWE-89, affected version ranges, SSVC, references, dates	Primary — CNA/ADP CVE record (JSON)	CVE.org / MITRE CVE Services API	https://cveawg.mitre.org/api/cve/CVE-2026-72898
2	CISA KEV entry: dateAdded, dueDate, shortDescription, requiredAction, knownRansomwareCampaignUse, cwes	Primary — CISA official KEV feed (JSON)	CISA	https://www.cisa.gov/sites/default/files/feeds/known_expl...
3	CISA KEV catalog (web)	Primary — government advisory	CISA	https://www.cisa.gov/known-exploited-vulnerabilities-cata...
4	BOD 26-04 directive text and risk criteria	Primary — CISA binding operational directive	CISA	https://www.cisa.gov/news-events/directives/bod-26-04-pri...
5	Official vendor advisory: CVE-2026-72898, root-cause summary, remediation steps, fixed versions	Primary — vendor security advisory	GitHub / Metabase	https://github.com/metabase/metabase/security/advisories/...
6	Official vendor advisory: CVE-2026-72899 (public dashboard SQLi), CVSS 9.6, remediation	Primary — vendor security advisory	GitHub / Metabase	https://github.com/metabase/metabase/security/advisories/...

#	Claim / Fact	Evidence type	Source	URL
7	Official vendor advisory: CVE-2026-72900 (cross-user data leak), CVSS 6.5, cleartext-credential caveat	Primary — vendor security advisory	GitHub / Metabase	https://github.com/metabase/metabase/security/advisories/...
8	Vendor's August 11 hardening release: 10 categories of fixes, HoneySQL CVE-2026-61620 dependency bump	Primary — vendor security advisory	GitHub / Metabase	https://github.com/metabase/metabase/security/advisories/...
9	Full list of historical Metabase security advisories (2022–2026), used for §3.3 pattern analysis	Primary — vendor advisory database	GitHub / Metabase	https://github.com/metabase/metabase/security/advisories
10	Vendor's own incident narrative, timeline, log-detection signature, fixed versions, Cloud-vs-self-hosted guidance	Primary — vendor blog/advisory	Metabase	https://www.metabase.com/blog/security-update
11	Technical root-cause analysis: Clojure `merge`, HoneySQL `:raw`, exposure statistics (13%/25%/~2,500 Shodan), PoC timeline	Primary technical analysis — cloud security vendor	Wiz	https://www.wiz.io/blog/inside-the-metabase-sqli-exploite...
12	Independent technical analysis: CWE-89 root cause framing, CVSS vector, detection/audit guidance	Primary technical analysis — security consultancy	Bishop Fox	https://bishopfox.com/blog/critical-sql-injection-in-meta...
13	Asset-discovery guidance and version matrix for defenders	Primary technical guidance — attack surface management vendor	runZero	https://www.runzero.com/blog/metabase/
14	Passive version-fingerprinting technique (`window.MetabaseBootstrap`), CVSS vector confirmation	Technical analysis — vulnerability intelligence vendor	IONIX	https://www.ionix.io/threat-center/cve-2026-72898/
15	Detailed detection/mitigation guide: log signatures, post-exploitation process indicators, exfiltration thresholds, interim network isolation steps	Secondary technical guidance	Security Arsenal	https://securityarsenal.com/blog/cve-2026-72898-metabase-...

#	Claim / Fact	Evidence type	Source	URL
16	n8n's own breach disclosure: 136 records, 5 with bcrypt hashes, historical plaintext-password bug, DPO/regulator notification	Primary — victim disclosure	n8n	https://blog.n8n.io/metabase-security-incident-update/
17	Checkly's own breach disclosure: 26-minute read-only session, scope, credential-rotation guidance	Primary — victim disclosure	ChecklyHQ	https://www.checklyhq.com/blog/metabase-security-incident/
18	Anaconda/Kilo Code's own breach disclosure: Slack token exposure, user data exposure, token invalidation	Primary — victim disclosure	Anaconda	https://www.anaconda.com/blog/metabase-incident-impacting...
19	Framework's breach disclosure details and timeline (notified 9am Pacific Aug 6)	Secondary reporting quoting primary disclosure	TechCrunch	https://techcrunch.com/2026/08/07/computer-maker-framewor...
20	Framework breach follow-up reporting	Secondary reporting	The Register	https://www.theregister.com/personal-tech/2026/08/10/fram...
21	Consolidated reporting: five named victims (Framework, Tally, n8n, Kilo Code, ChecklyHQ), exposure statistics, log-pattern IOC	Secondary reporting, aggregating primary disclosures	Help Net Security	https://www.helpnetsecurity.com/2026/08/10/metabase-zero-...
22	Consolidated reporting: Framework and Tally disclosures, attack-pattern IOC, timeline	Secondary reporting	BleepingComputer	https://www.bleepingcomputer.com/news/security/framework-...
23	Consolidated reporting with named expert quotes (David Shipley, Beauceron Security; Scott Miserendino, DataBee)	Secondary reporting with named expert commentary	CSO Online	https://www.csoonline.com/article/4208307/metabase-sqli-e...
24	Technical summary and IOC confirmation	Secondary reporting	The Hacker News	https://thehackernews.com/2026/08/metabase-zero-day-explo...
25	Technical summary, log-signature quote from Metabase, timeline	Secondary reporting	Security Affairs	https://securityaffairs.com/196874/hacking/metabase-zero-...
26	"Five companies breached" framing and credential-exposure summary	Secondary reporting	Tech Times	https://www.techtimes.com/articles/324060/20260812/metaba...

#	Claim / Fact	Evidence type	Source	URL
27	CISA's own alert on the August 11, 2026 KEV batch (Cisco ASA/FTD, Windows AFD, and Metabase)	Primary — government alert	CISA	https://www.cisa.gov/news-events/alerts/2026/08/11/cisa-a...

Note on sourcing methodology: every version-matrix figure, CVSS vector, and vendor-remediation quote in this report was cross-checked directly against the GitHub Security Advisory API (`api.github.com/repos/metabase/metabase/security-advisories/...`) and CISA's/MITRE's own machine-readable feeds (KEV JSON and CVE Services API), rather than relying solely on secondary press summaries, specifically to avoid propagating any third-party transcription errors. Where a fact could only be found in secondary reporting (e.g., named expert quotes, some victim-notification timing details), it is marked as such in the table above.