

# Incident Response Plan

How a security or availability incident is identified, contained, and reviewed.

Version 1.0, effective 2026-08. Owner: Himaja Prabhala. Next scheduled review: 2027-02.

---

## 1. Purpose and scope

This plan covers TRACE Nexus and the infrastructure it runs on: the application server, the internal detection-engine service, the database and its row-level security policies, and the network path in front of all three (Cloudflare Tunnel and Cloudflare Access). It applies to any event that compromises the confidentiality, integrity, or availability of case data or the systems that process it, whether caused by external attack, misconfiguration, or human error.

## 2. Current operating reality

TRACE Nexus is currently operated and maintained by one person. This plan is written against that reality rather than assuming a team that does not yet exist: every responsibility below currently falls to that one person, and "escalation" means a documented pause to think rather than a handoff to a separate on-call engineer. This section should be the first thing revised if and when the team grows.

## 3. Severity classification

| Severity | Definition                                                                                                                      | Example                                                                                           |
|----------|---------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| Critical | Case data or credentials exposed to unauthorised access, or the system is completely unavailable.                               | An access-control change accidentally removes authentication from a route that should require it. |
| High     | A control that should prevent unauthorised access is weakened or bypassed, but no confirmed exposure of real data has occurred. | A single defensive layer (of three) fails or is misconfigured, while the others still hold.       |
| Medium   | A functional defect that does not itself expose data but could mask a real problem or degrade evidentiary integrity.            | A finding fails to persist correctly, or the audit log silently fails to record an action.        |
| Low      | An operational issue with no security or data-integrity impact.                                                                 | A slow page load, a cosmetic UI bug.                                                              |

## 4. Detection sources

Incidents are expected to surface through one of the following, in order of how much they should be trusted:

- Direct verification against the running system (a curl check against the actual origin server), not the dashboard UI alone. A configuration screen can look correct while the live behaviour is wrong.

- The append-only audit log (audit\_log table), which records every upload, view, edit, and approval and cannot be alt+deleted, edited, or backdated by design, no update or delete policy exists for it at the database level.
- systemd service status and VM resource monitoring (memory, load average) for availability issues.
- Cloudflare Access and tunnel logs for edge-level authentication anomalies.

## **5. Response procedure**

### **1. Identify**

Confirm the issue is real by checking the live system directly, not just the administrative interface. Record what is actually observed, with timestamps, before taking any corrective action.

### **2. Contain**

Take the fastest action that stops further exposure, even if it is not yet the correct long-term fix. For an access-control gap, this typically means reverting the specific change that caused it, not a broader shutdown.

### **3. Eradicate**

Identify and fix the root cause, not just the symptom. A change that accidentally widened access should be traced to why it was possible to make that change, not only reverted.

### **4. Recover**

Confirm the fix directly, the same way the issue was first confirmed: verify the live system, not the dashboard. Confirm every unrelated control that could have been affected is still intact before considering the incident closed.

### **5. Review**

Record what happened, why, and what changed as a result, in this document or a linked record. A fix that isn't written down is a fix that can be silently reintroduced later.

## **6. A real incident, worked through this process**

This plan is not purely theoretical. During this system's own development, a change intended to make one public page reachable without login was applied to the wrong underlying configuration, briefly removing the authentication requirement from the entire application rather than the single page intended. It was caught within minutes by verifying the live system directly rather than trusting the configuration screen, contained by reverting the specific change, and closed only after independently re-verifying that every previously protected route was protected again. The lesson from that incident, verify the live system after every access-control change, not just the screen that made it, is now standing practice, and is the reason step 1 and step 4 above both insist on direct verification rather than trusting the interface.

## **7. Notification**

For a Critical or High severity incident involving any real (non-synthetic) client data, affected clients are notified as soon as the facts are established, not after the full investigation completes. Until this system has real clients, this section is a standing commitment for when it does, not a retrospective claim about incidents that have already happened, the incident described above involved no real client data.

## 8. Review cadence

This plan is reviewed at minimum every six months, and immediately after any Critical or High severity incident, whichever comes first.

---

TRACE Nexus. Internal governance document. Not for external distribution without review. Copyright 2026 Himaja Prabhala. All rights reserved.