Bug Triage & Workflow

Post-release issues for Team API Keys & Proxy Pools

1. Issue Prioritization

#	Issue	Priority	Why it matters
1	Dashboard never loads for accounts with 10k+ keys	**P0**	Enterprise customers literally can't manage their keys—scrapers are running blind. High churn risk if not fixed fast.
2	Activity log stays blank after revoking a key	**P0**	No visibility into revocation means security teams can't verify the key is actually dead. Compliance nightmare.
4	Revoked keys still work from some countries (Residential)	**P0**	Worst one here—customers think keys are revoked but they're still active in certain regions. Major trust issue + abuse risk.
5	E2E tests timing out randomly in CI	**P1**	Slowing down our ability to ship fixes confidently. Needs attention before it blocks the next hotfix.
3	Button says "Create new keyy" (typo)	**P3**	Embarrassing but harmless—doesn't block actual work. Can fix when we're touching that section anyway.

2. Sample Bug Ticket

Revoked API keys still functional on Residential proxies from certain geos

Severity:

Critical

Priority:

P0

Labels:

security, customer-facing, regression, residential-proxies

Customer Impact:

Paying customers can't reliably revoke keys. Traffic continues flowing from specific countries even after revocation, creating potential for account abuse, data leaks, billing overages, and loss of trust in our security controls.

Environment:

Production – Residential Proxies endpoint (residential.proxy.com). Confirmed from German and Brazilian IPs, possibly others.

Affected Version:

v1.2.0 (Team API Keys & Proxy Pools release)

Steps to Reproduce:

1. Login to paid account with Residential access
2. Create new API key via Dashboard → API Keys
3. Send test request through key (works fine)
4. Revoke the key in dashboard
5. Retry same request from German or Brazilian IP (via VPN or actual exit node)

Expected:

All requests with revoked key should get 401 Unauthorized regardless of country.

Actual:

Requests from Germany, Brazil (and possibly other non-US regions) still get 200 OK with full proxy functionality. US IPs correctly blocked.

Sample API Call (should fail but doesn't):

```
curl -v https://ipinfo.io/json \ -x http://residential.proxy.com:8080 \ -H
"Authorization: Bearer sk_live_revoked_9x8y7z6w5v4u3t2s1r" \ -H "Proxy-Country: DE"
```

Attachments:

- revoked_key_dashboard.png – screenshot showing key status = Revoked
- curl_success_after_revoke.txt – verbose output showing successful request after revoke
- proxy_node_eu.log – EU region logs missing revocation entry for this key
- customer_example_redacted.txt – customer's success response 8s after revoke

Notes:

- 100% repro rate from affected geos within 10s of revocation
- Probably revocation not propagating to all residential node regions, or geo-routing bypasses auth layer

- Seems related to Bug #2 (missing activity log entries)

3. Bug Workflow: Support → QA → Dev → Release

Step	Owner	What Happens	When	Who's Notified
1	Support	Customer reports issue → Support creates ticket using template (repro steps, impact, logs) → posts link in #qa-triage Slack	Same day, ideally under 2hrs	QA Lead (ping)
2	QA Lead	Try to reproduce at same scale → record Loom walkthrough → assign priority (P0/P1/P2) → add hypothesis to ticket	Within 4 hours	Support (auto-watches)
3	QA → Dev	P0/P1: immediately tag @dev-lead + on-call engineer, move to "Ready". P2: add to next sprint	Instant for P0/P1	Dev lead + on-call
4	Dev	Pick up ticket (with Loom + clear repro) → implement fix → open PR	Same or next day	QA (PR reviewer)
5	QA	Test fix in staging at real scale → run quick regression suite → approve or send back	Before merge	Dev (if rejected)
6	Release	P0/P1 → immediate hotfix (with canary if needed). P2 → next regular deploy	ASAP for hotfix	—
7	QA	Verify in prod → comment on ticket "@support fixed in prod, have customer hard-refresh"	Right after deploy	Support + customer
8	QA	Weekly summary to Founder/Product: customer-facing bugs only, priorities, patterns (especially AI-related issues)	Every Friday	Founder + Product

4. AI-Assisted Code Issues

AI tools are great for speed but they default to the "happy path" examples from their training data. They rarely think about edge cases like handling 10k rows or distributed systems across 50 countries. Here's what probably went wrong:

Issue #1: Dashboard hangs with 10k+ keys
Dev probably asked AI for "list with pagination" and got basic `SELECT * WHERE team_id = ?`

with client-side pagination or no limits. Works fine with 50 test keys, completely dies with real customer data. AI just doesn't consider scale unless you explicitly tell it to.

Issue #4: Revoked keys work in some countries
AI generated clean `UPDATE revoked = true` but forgot about invalidating distributed edge caches. Most examples online are single-server setups, so cache invalidation across regions just doesn't show up in typical prompts.

Issue #5: Flaky test timeouts
E2E tests probably copied some example with `timeout=15s`. Nobody told the AI that revocation now triggers async jobs that take 10-20s under load. Magic numbers without context = flaky tests.

Issue #3: "keyy" typo
Classic hallucination. AI was generating UI components and just made up the button text. Happens when prompts aren't specific about exact copy.

Prevention Strategy

I'm not banning AI—just making careless usage painful so people get better at it:

1. Mandatory AI disclosure on PRs
Every PR using AI needs an "AI-Generated" label plus the actual prompt pasted in comments. No label = instant reject. This forces better prompts ("handle 20k rows with cursor pagination, invalidate edge caches") instead of vague requests. Takes 5 seconds to check, prevents days of debugging.

2. Standard checklist for AI-heavy PRs
I comment the same 4 questions on every AI PR:

- Does this handle >1k rows or multiple regions?
- Did you explicitly ask for pagination/cache invalidation in the prompt?
- Did you test on a real-scale account?
- Any hardcoded timeouts?

One "no" = reject with link to our large-account test dataset. Public rejections make people fix their habits fast.

3. Nightly CI with 15k-key test account
Costs almost nothing, catches "works on my machine" issues before they hit customers.

Fails loudly when someone sneaks in unscalable code.

4. Bug retros track AI contribution
Every customer bug gets tagged "AI-related: Yes/No" with a visible scoreboard. Nobody wants their name next to multiple P0s caused by lazy prompts.

> **End result:** Still shipping fast with AI, but the scale-related disasters drop to near zero. Team learns to write better prompts in about a week, and I barely have to intervene after the first few PR rejections.

5. Hidden Impact: Bug #1 Cascade

The dashboard hang (Bug #1) is actually way worse than it looks on the surface. Here's what's really happening:

That unpaginated query is doing a full table scan and holding DB connections open for 10-20 seconds per request. When big customers leave the dashboard tab open (which they do), those connections never close. The shared connection pool gets drained.

What breaks as a result:

- Proxy auth starts timing out → customers see random 502/503 errors
- New key creation fails intermittently
- Activity logs get delayed or dropped entirely → makes Bug #2 and #4 way harder to debug because we're missing audit trail

So we end up spending weeks chasing "intermittent proxy issues" and "missing logs" when the actual root cause is just people trying to view their key list. I've seen this pattern before—similar unpaginated admin endpoints ate 40-60% of our total DB capacity in a previous role.

Fix the pagination, and suddenly all these "unrelated" issues disappear.

> **Lesson:** Performance bugs that only show up at scale aren't just UI problems—they're infrastructure threats that cascade into other systems. Always test with real customer data sizes.
