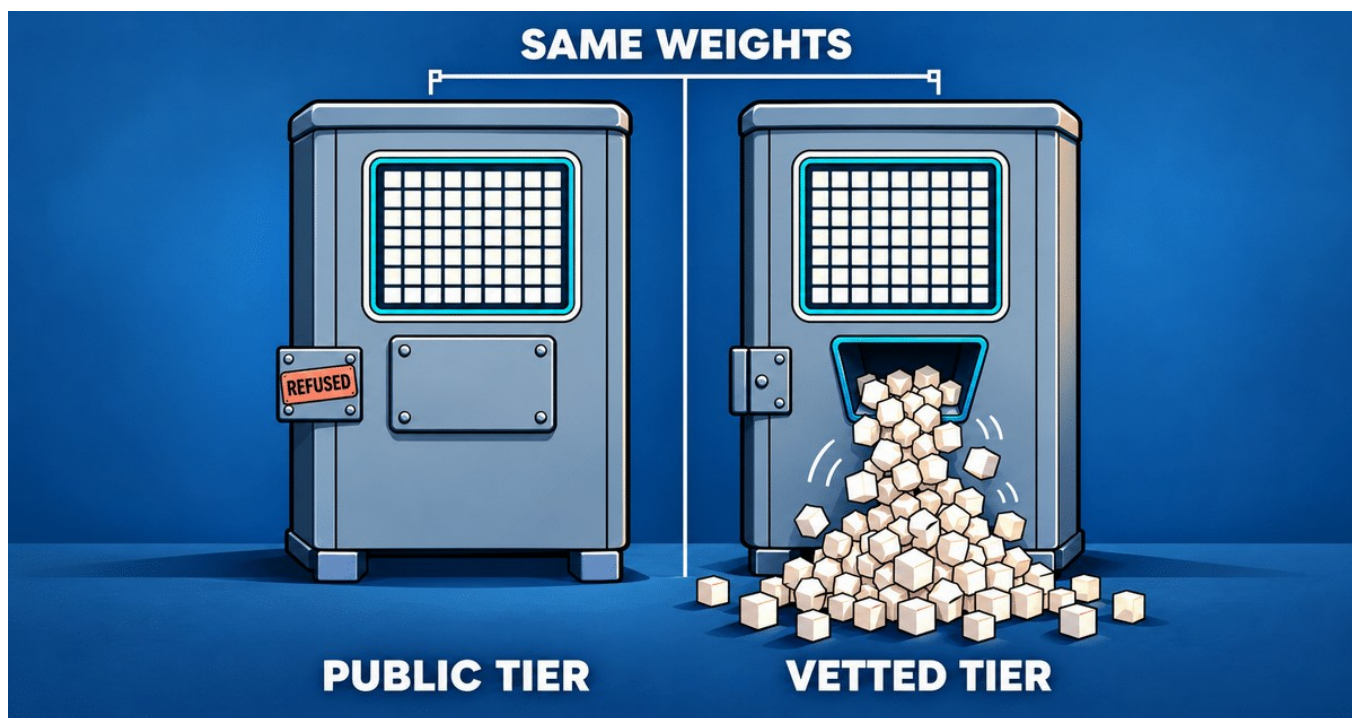


After Coldcard: AI Security Depends on Access

Julien Laurenceau | Published 2026-08-07 | Updated 2026-10-22

<https://pepitedata.com/coldcard-ai-vulnerability-scan-censored-models/>

PepiteData | LinkedIn: Julien Laurenceau | X: @Julien_Lau



Same model, different access policy: a conceptual cartoon of a blocked public output and an available vetted output. This is not a universal claim about provider tiers.

AI-assisted security review depends on the model, the access policy and the work needed to validate its output. The Bitcoin Red Team campaign after the Coldcard incident made those dependencies visible. Its reported findings were substantial, but they were not a controlled model benchmark or a count of independently confirmed bugs.

I would put refusal behavior on the same dependency list as quotas, data handling and provider availability. A model that can solve a task in a demonstration may still be unavailable for that task under your account's access conditions. That is a procurement and operations problem worth testing before an incident.

What did the Coldcard incident expose?

The July 2026 Coldcard incident concerned weakened seed generation, according to the vendor's current security record. Attackers could regenerate corresponding private keys offline; the devices were not remotely taken over. The distinction matters because a review of cryptographic algorithms alone does not validate the firmware build or the actual randomness path used by a released device.

In its [August 5 analysis](#), TRM Labs described a build configuration error and reported a preliminary tally near 1,816 BTC, approximately \$116 million at the time. That is an attributed historical estimate, not a current loss total or an independent audit by me.

The [Coldcard security status page](#), checked for this October 22 revision, records fixed releases and targeted independent checks. It explicitly distinguishes correcting future seed generation from repairing an existing affected seed, and does not claim a complete audit of every firmware binary. For device-specific action, use that living advisory rather than this infrastructure article.

The broader review question is concrete: does the released artifact actually use the implementation the design assumes? Source review, build verification and runtime checks answer different parts of that question. None makes the others redundant.

What did the Bitcoin Red Team report?

The Bitcoin Red Team reported 6,700 findings across 425 projects after 55 hours, including 1,029 classified as high or critical. These were campaign figures published by its organizer, not independently adjudicated vulnerability counts. The update described reproduction work and outreach bottlenecks, so the numbers should be read as a snapshot of an active review process.

In [calle's August 6 update](#), the team said it was trying to reproduce critical findings in local regression-test environments before reporting them. The thread also identified difficulty reaching maintainers. Counting generated reports misses both jobs.

The effort continued beyond that early snapshot. [OpenSats announced priority red-team support on August 6](#), including reimbursement of LLM token costs. Its [October 12 grant announcement](#) describes support for researchers conducting AI-assisted scanning and related security work. Funding activity corroborates an ongoing effort; it does not independently validate every reported finding.



A finding enters a queue. Reproduction and review determine which reports become confirmed vulnerabilities. The illustration encodes no measured conversion rate.

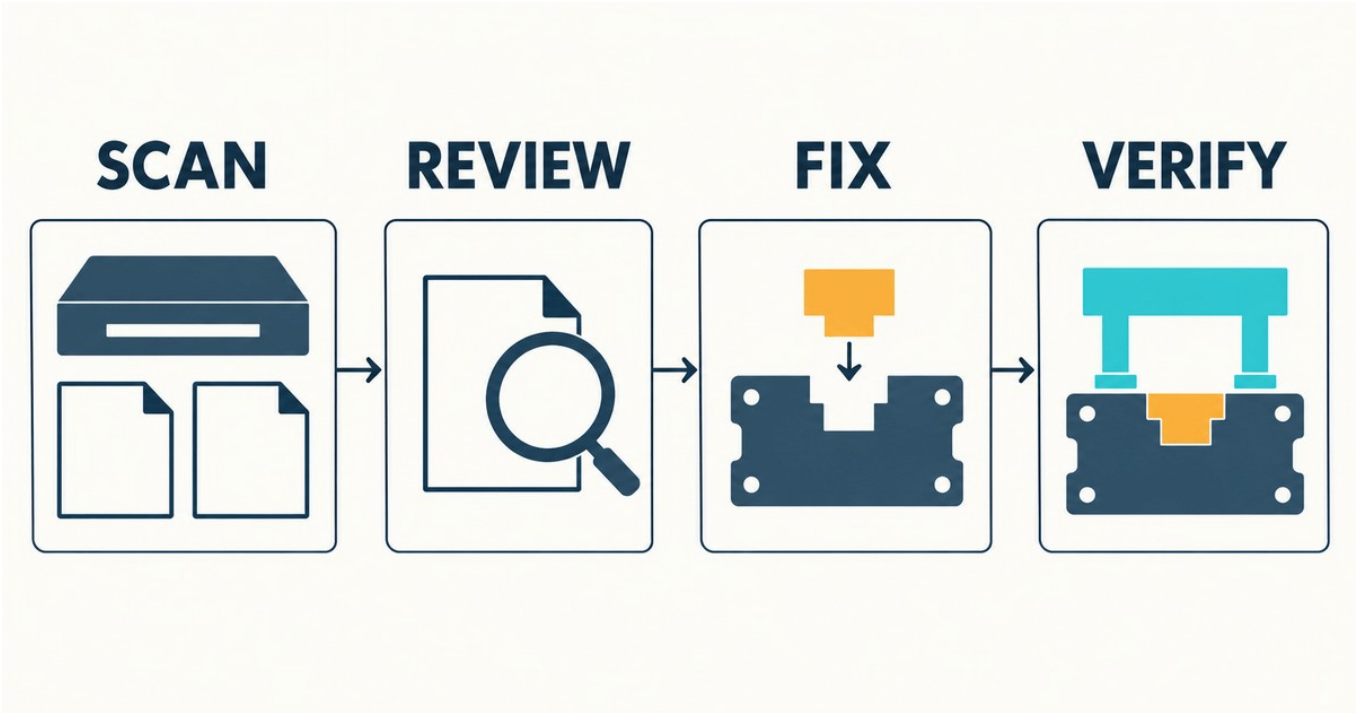
A finding is not a confirmed vulnerability

An AI finding becomes useful security evidence when a reviewer establishes the affected path, necessary conditions and actual impact. Severity labels should follow that evidence. Deduplication, maintainer review and verification of a fix are separate stages. A high report count can increase review workload without proving a corresponding increase in confirmed or remediated vulnerabilities.

Stage	Evidence needed	What the count means
Generated finding	A report with a specific code or configuration claim	Candidate issue, possibly incorrect or duplicated
Reproduced issue	A repeatable test in an authorized environment	Observed behavior under documented conditions
Confirmed vulnerability	Impact and affected versions reviewed	A security issue with a supported scope
Verified remediation	Fix review and a regression test	The tested failure is addressed in the checked release

This is my review framework, not a reconstruction of the campaign's unpublished triage records. A useful dashboard would keep those counts separate and track the age of unresolved reports. I would also measure reviewer time and rejected findings before describing the token bill as the cost of an audit.

Disclosure adds another constraint: a valid report needs a responsible destination. Maintain a private reporting contact and a process for acknowledgment and follow-up. Publishing raw suspected vulnerabilities to prove scanning productivity can create exposure before maintainers have checked them.



Keep generation, validation and remediation separate. Completion at one stage does not prove completion at the next.

Did public models really find nothing?

The campaign's early model-access statements do not establish that public US models lack vulnerability-finding capability. In the same August 6 discussion that criticized their contribution, the organizer said team members with cyber-tier access had found vulnerabilities. That qualification changes the interpretation: the account and access conditions belong in the comparison alongside the model.

The [initial post](#) names Kimi K3 and Qwen 3.8 and criticizes the lack of US frontier-model findings. The [immediate follow-up](#) says members with GPT cyber access did find vulnerabilities. I would not turn that exchange into a league table or the claim that every public model refused every defensive task.

A defensible comparison would record the same repository revision, task, tools, prompts, access tier and review criteria. It would separate a policy refusal from an incorrect answer, a tool failure and an actual absence of findings. The campaign reports do not supply that controlled comparison.

Access policy is a production dependency

A security workflow can fail because access is restricted even when the underlying model can perform the task. Provider safeguards, account approval and routing affect the service a team actually receives. Evaluate those conditions on the intended product surface and account. A model name or a successful demonstration under another organization's access is insufficient.

Anthropic's [June 30 Fable 5 update](#) describes safeguards that can block benign security work and reroute blocked requests to Opus 4.8. It says Fable 5 and Mythos 5 share an underlying model, while their safeguards and distribution differ. This is a documented example, not evidence that all providers ship identical weights across tiers.

The current [official OpenAI cybersecurity documentation](#), checked October 22, describes Trusted Access for Cyber as a reviewed program. Approval is scoped to the authorized person or service, organization or workspace, model and product surface. It does not automatically grant specialist access or Zero Data Retention, and does not configure the tools or authorized engagement scope.

Dependency	What to verify	Failure to plan for
Access approval	Exact account, organization, surface and authorized use	A tested workflow is unavailable to its production identity
Routing and safeguards	Refusal, fallback and error behavior on approved tasks	A request stops or reaches a different model
Data handling	Permitted inputs, retention and destination	Sensitive code reaches an unsuitable service
Fallback path	Quality, capacity, tools and operator procedure	An alternative exists on paper but cannot finish the work

The table is a procurement checklist, not a promise of eligibility. A legitimate defensive purpose still needs authorization and appropriate controls. Plan for denied or interrupted work; do not build reliability around bypassing a provider's safeguards.

Does local inference solve the dependency?

Local inference gives an operator more control over model artifacts and serving configuration, but does not guarantee correctness, stable behavior or sufficient capability. Model weights can contain refusal behavior and bias. Runtime versions, prompts, tool permissions and retrieval data also affect results. A usable fallback requires testing the whole workflow, not merely downloading weights.

I would retain checksums and versions for the model, tokenizer, runtime and relevant configuration, then rerun representative tests after changes. Keeping the weights unchanged does not freeze the surrounding application. An alternative provider running an open-weight model also remains a hosted dependency.

The evaluation approach in [local LLMs and political-bias testing](#) makes a related distinction: control over execution is useful, while claims about model behavior still need scoped evidence. For the full dependency map, use the [AI production supply-chain checklist](#).

What should a platform team test first?

A platform team should test a small set of authorized security workflows using the identity, tools and data controls intended for production. Record successful completion, refusals, routing changes and review effort separately. Define a manual or alternative-model procedure for blocked work. A fallback is credible only after it completes representative tasks within the required constraints.

1. Select representative defensive tasks and document the systems and actions that are authorized.
2. Run them under the intended account and product surface; record model and runtime versions where available.
3. Review the evidence and classify failures: policy refusal, access error, tool failure or quality failure.
4. Rehearse the approved fallback, including human review, data boundaries and the time needed to finish.
5. Repeat the checks after material changes and retain enough non-sensitive evidence to diagnose regressions.

This belongs within [AI infrastructure](#) and [SRE](#), rather than being a one-off model selection exercise. If an upcoming deployment depends on those assumptions, my [Flash Architecture Review](#) can examine the dependency and fallback design before it becomes operationally critical.

FAQ: AI-assisted security review

AI-assisted security review needs both access controls and evidence checks. The questions below separate reported campaign output from validated security results, and model capabilities from the conditions under which a team can use them. The answers apply to authorized defensive work; they do not expand permission to test systems or disclose sensitive findings.

Are the 1,029 findings confirmed bugs?

The 1,029 figure is the campaign organizer's high-or-critical classification at the 55-hour snapshot. It should not be presented as an independently confirmed bug count or a count of fixes verified in released software.

Does a refusal prove a model cannot do the task?

A refusal can result from policy or access controls rather than a lack of capability. It still affects operational usefulness. Evaluate refusals separately from wrong answers, tool failures and incomplete investigations.

Are open-weight models free of safeguards or bias?

Open weights do not guarantee the absence of refusal behavior or bias. Self-hosting increases control over artifacts and execution, but the model and the complete serving workflow still need evaluation.

Should a blocked security task be retried elsewhere?

Use an approved alternative only when the task remains authorized and the destination meets the required data and security controls. Otherwise stop and escalate to the responsible operator. A fallback must preserve scope and confidentiality.

Sources and revision scope

The campaign posts provide self-reported activity and access observations. Vendor documentation describes particular safeguards or remediation mechanisms, rather than independently establishing model rankings or absence of defects. The tables and deployment procedure are my engineering

recommendations. This article reports public information and contains no customer code, customer identifiers or unpublished vulnerability details.

- [COLDCARD, current security status](#), living incident and remediation record, checked October 22, 2026.
- [TRM Labs, incident analysis](#), August 5, 2026.
- [calle, 55-hour campaign update](#), August 6, 2026.
- [calle, model-access observation and qualification about cyber-tier findings](#), August 6, 2026.
- [OpenSats, Code RED](#), August 6, and [first red-team grants](#), October 12, 2026.
- [Anthropic, Redeploying Fable 5](#), June 30, 2026.
- [OpenAI, Cybersecurity checks](#), current documentation checked October 22, 2026.

No affiliation with or endorsement by the cited organizations is implied.

Updated October 22, 2026. Replaced the categorical model-ranking headline, distinguished reported findings from validated vulnerabilities, added current access documentation and the vendor's remediation record, and rebuilt the image pack. The URL and original publication date are unchanged.

Related posts

This article focuses on access-policy dependencies in AI-assisted security review. The AI infrastructure and SRE guide covers the broader operating model, while the related articles examine supply-chain control and local evaluation. Together they help separate the model artifact, its hosted service and the workflow that a team actually depends on.

- [AI infrastructure and SRE: running models in production](#)
- [The AI supply chain as a production dependency](#)
- [Local LLMs: control and evidence about model behavior](#)

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