ProjectX

PROVING ACCESS DECISIONS TO AN AUDITOR · USE-CASE BRIEF

Show an auditor who could do what, who did, and why it was allowed.

Every allow and deny across every pillar in one hash-chained audit, a reason and policy on every denial, reverse and forward entitlement queries, exports in the formats auditors use, and a separate record of who looked at footage.

The situation

The audit request asks who had access to the finance data last quarter and why. The answer is spread over five consoles, three log formats and a spreadsheet of role assignments nobody trusts.

How it works, step by step

01 **Record every decision** · Trust and security
Allows and denies from every service land in one schema, hash-chained with a verification endpoint.

02 **Explain each refusal** · Security
Every denial returns a reason and the policy that made it.

03 **Answer “who can” and “what can”** · Security
Ask who can act on any resource, or what any principal can do, and simulate a policy change before it ships.

04 **Hand over the evidence** · Trust and security
Export to CSV, Parquet, OCSF or CEF, or run the access review, privileged action and data access reports.

05 **Show who looked at footage** · Investigations
Search, view, export and face-match audits are separately reportable, and cannot be disabled by a tenant administrator.

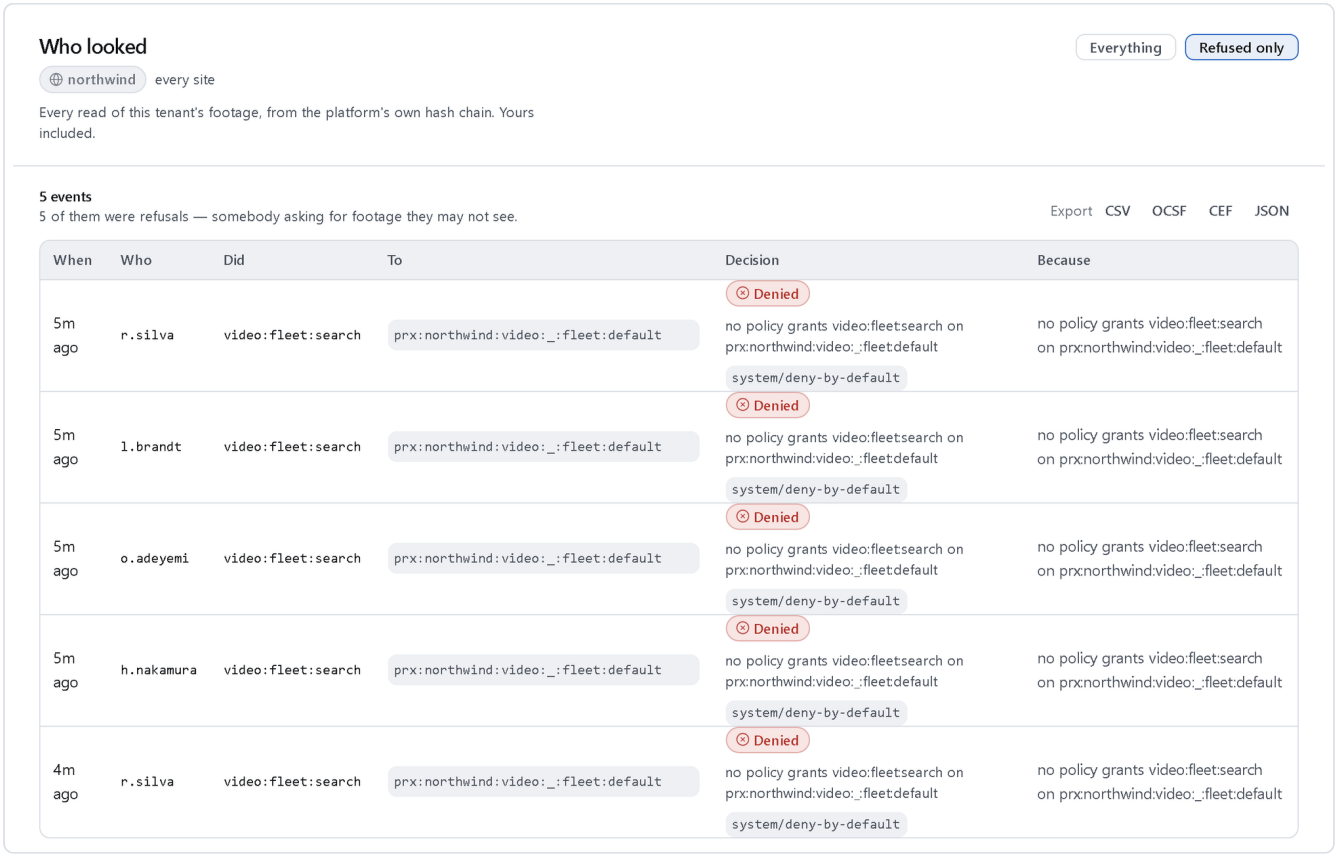
What you get

01
One record, provably complete
A single hash chain across pillars, with a verification endpoint an auditor can use.

02
Every no has a reason
Denials carry their reason and policy, so a refusal is explainable.

03
Entitlements on demand
Who-can and what-can queries replace the spreadsheet of role assignments.

In the console



Every read of footage, including the ones policy refused. Screens shown with demo data.

Questions buyers ask

Can the audit be tampered with?

Events are hash-chained with a verification endpoint, and the chain can optionally be anchored externally.

How long is audit kept?

By a retention policy with legal minimums and legal hold.