ONAM SECURITY

TECHNICAL WHITEPAPER · 03

One graph, one data model

Why every engine writes the same finding contract into one store — and what that makes possible that a drawer of tools cannot.

One canonical store

Deterministic finding IDs

Per-tenant property graph

Typed abuse edges

29 engines

THE ARGUMENT, IN ORDER

READS WITH 01 Attack-path methodology

00 Executive summary

Consolidation stops being a slide when it becomes a data model.

Most security platforms are a drawer of tools that each keep their own findings and their own view of the world. Onam is built the opposite way: every engine, for every cloud, writes the same finding contract into one shared store, and everything reasons over one property graph. This paper explains that design and why it is the thing that makes cross-cloud attack paths, priced risk and painless extensibility possible.

The architectural claim is specific and testable: **there are no per-engine report tables to reconcile.** A gateway reads only from one canonical findings store; a per-tenant graph turns those findings and the assets they touch into nodes and typed edges. Adding a capability does not add a silo — it adds another writer to the same store.

> **Why it matters to a buyer.** Posture, identity, data, workloads, SaaS and runtime detection share one schema, so a finding in one domain can form part of an attack path that ends in another — which is impossible when each tool hoards its own results.

01 Ingestion into a normalised inventory

Two tables come out of ingestion, and every capability in the platform depends on them.

Each cloud connects read-only and agentless (Whitepaper 01, §02). The pipeline runs four stages — enumeration → enrichment → normalisation → drift — and writes two things every capability depends on: an asset inventory and an asset-relationship table.

7 clouds plus SaaS collapse into one normalised identity space, covering 549 cloud services and 11,433 rule definitions.

02 The canonical finding contract

Consolidation lives or dies on the schema.

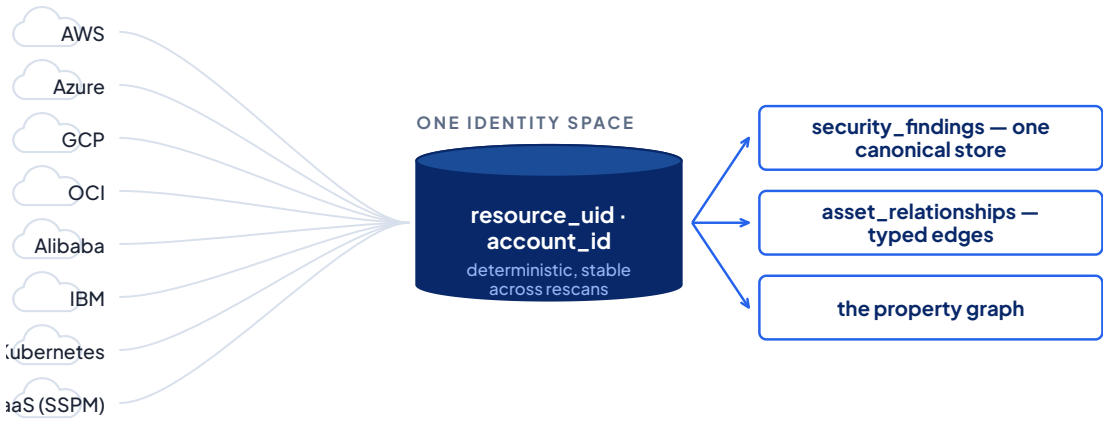

FIGURE 1 — MANY CLOUDS, ONE NORMALISED STORE

The narrowing is the architecture. Cross-cloud attack paths are only visible because both ends of the route live in the same store.

Onam requires every engine to emit the same standardised finding, keyed consistently, so one table can hold results from posture, identity, data, workload, code and SaaS engines without translation.

Field	Canonicalises	Why it matters
resource_uid	ARN vs OCID vs self-link	One identity for an asset across clouds
account_id	account vs subscription vs project	Ownership normalised across providers
finding_id	deterministic hash	Same issue → same ID on every rescan; stable history
tenant_id	not-null isolation key	Every row is tenant-scoped by construction
provider_metadata	provider-specific detail	Nothing is lost; specifics travel with the finding

Because `finding_id` is deterministic, remediation tracking is reliable: a fixed issue auto-resolves on the next PASS and auto-closes after 30 days unseen, without a human reconciling IDs by hand.

03 The property graph

The cloud documents its relationships. The graph adds the ones an attacker would use.

On top of the findings and relationships, Onam builds a per-tenant property graph. Assets are nodes; edges are typed — containment, attachment, IAM trust and network reachability — and catalog-driven derivers add the abuse edges an attacker would use.

The derivers are the important part. A cloud provider documents the relationships it intends; an attacker uses the ones that exist. The graph has to hold both.

04 Engines as writers, not silos

29 engines write into the same store. That is the extensibility claim: a new capability is a new writer, not a new database, a new schema and a new reconciliation job. It is also why a posture finding and a runtime detection can appear on the same path — they were never in separate systems to begin with.

> **The honest limit.** One shared schema is a constraint as well as a gift. An engine whose output does not fit the contract has to be modelled to fit it, and that is real work. We take that cost deliberately, because the alternative — per-engine tables — is the thing that makes cross-domain reasoning impossible later.

05 What this architecture buys

- **Cross-cloud attack paths.** A route from one provider to another is only visible when both are described in one vocabulary.
- **Priced risk.** FAIR needs consistent asset identity and data classification to compute magnitude; the shared schema supplies both.
- **Stable history.** Deterministic IDs mean trend lines and remediation tracking are real, not approximate.
- **Extensibility.** Adding an engine adds coverage, not another silo to reconcile.

06 Summary

Adding a capability adds a writer to the same store — never another silo.

- One canonical findings store. No per-engine report tables.
- A standardised finding contract with deterministic IDs and mandatory tenant scoping.
- A per-tenant property graph with typed edges, plus derived abuse edges.
- 29 engines as writers into the same model — extensibility without silos.
- The constraint is real and accepted: fitting the contract is work, and worth it.