ONAM SECURITY

TECHNICAL WHITEPAPER · 01

How Onam finds the paths that matter

From read-only telemetry to a verified, MITRE-mapped, priced attack path — the method, end to end.

7 clouds, read-only

One property graph

5-domain verification

MITRE ATT&CK

Choke points

FAIR pricing

THE ARGUMENT, IN ORDER

READS WITH 02 Cloud risk in dollars · 03 One graph, one data model · 04 Security & trust

00 Executive summary

A finding is a fact. A verified path is a decision.

Every cloud scanner produces thousands of findings. The hard problem is not detection — it is knowing which handful of them combine into a route an attacker could actually walk to something that matters, and what that route is worth in dollars. Onam treats that as a graph-reasoning problem, solved the same way every time.

This paper describes the method end to end: how Onam ingests 7 clouds read-only, normalises everything into one property graph, derives the relationships an attacker would exploit, searches for routes from internet-reachable entry points to crown jewels, and — critically — verifies every edge across five independent security domains before it is ever called a confirmed path. Each hop is mapped to a MITRE ATT&CK technique; the full set of paths is reduced to its top five choke points; and the resulting exposure is priced with the FAIR model, so the output is a board-ready number rather than a severity label.

> **The claim, precisely.** Onam does not promise to block attacks at runtime. It promises something more useful for prioritisation: a reproducible, verified and priced map of the routes that exist in your cloud right now — computed from read-only telemetry, with a deterministic identity for every finding so results are stable across rescans.

01 The problem with a list

A modern multi-cloud estate generates findings faster than any team can trace them. The industry's default answer is severity: sort by Critical, work down the list. But severity is a property of a finding **in isolation**. It cannot tell you that a "medium" public endpoint, a "low" over-permissive role and an unencrypted data store are — together — a single four-hop route to your customer records.

Attackers do not read your list. They chain. A route is only as strong as its weakest link, and the link that matters is rarely the one with the scariest CVSS score. To prioritise like a defender who thinks like an attacker, you have to reason over the relationships between findings, not the findings themselves.

02 Ingestion — read-only, agentless, 7 clouds

Onam holds no standing credential in your cloud — only a reference it exchanges for a token that expires.

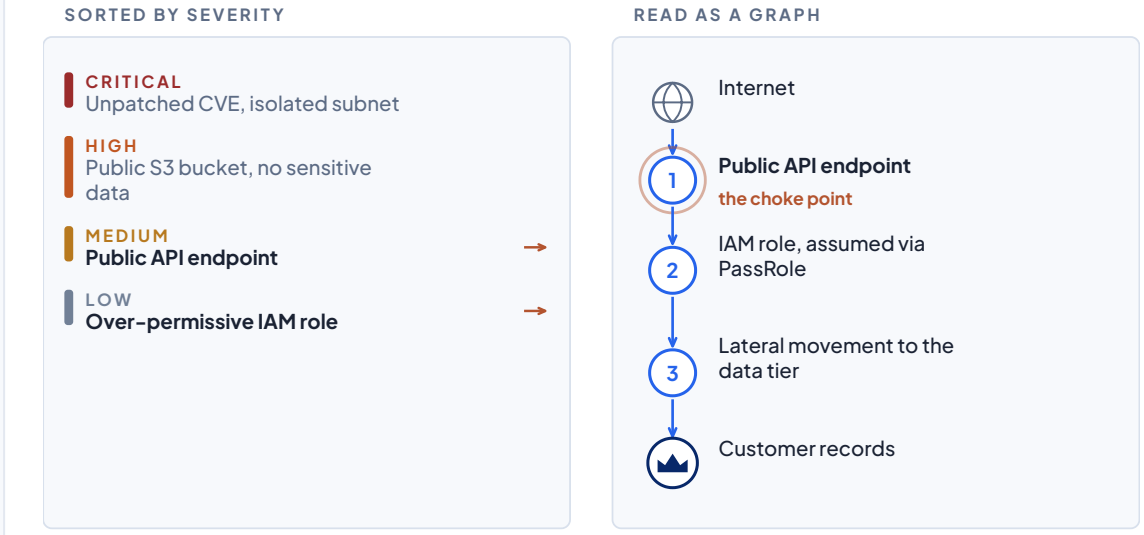

FIGURE 1 — THE SHIFT

The same four findings, read two ways. Severity scatters them; the graph connects them into one prioritisable route with a single highest-leverage fix. Illustrative.

The method starts with telemetry that is safe to collect. Onam connects to each cloud with the lowest standing privilege that permits a security read, and stores no long-lived credentials — only a reference to the role or principal, exchanged for short-lived tokens at scan time.

Cloud	Connection	Privilege posture
AWS	Cross-account IAM role via sts:AssumeRole, unique External ID per tenant	SecurityAudit + ReadOnlyAccess; STS credentials ≤ 60 min
Azure	Entra service principal	Reader + read-only Graph
GCP	Service account or keyless Workload Identity Federation	6 read-only roles
OCI	IAM user + signing key	inspect + read only
Alibaba / IBM	RAM read-only · Service ID	Read-only, exchanged for a short-lived Bearer token
Kubernetes	Read-only ServiceAccount	get / list / watch

Full detail of the trust model is in Whitepaper 04.

03 One data model — the property graph

Everything normalises into a single per-tenant property graph. Assets are nodes; edges are typed — containment, attachment, IAM trust and network reachability. Catalog-driven derivers then add the *abuse* edges: the relationships an attacker would actually use, as distinct from the ones the cloud provider documents.

This is the step that makes cross-cloud reasoning possible at all. A path that starts in one provider and ends in another is only visible when both are described in the same vocabulary. Architecture detail is in Whitepaper 03.

04 Search — BFS from entry to crown jewel

Search is breadth-first from internet-reachable entry points toward crown jewels — the assets a customer has designated as mattering. Breadth-first matters: it finds the *shortest* route first, and the shortest route is usually the one an attacker takes.

05 Verification — five domains before "confirmed"

A path that exists in the graph is a hypothesis, not a finding. Before Onam calls a path confirmed, every edge is verified across five independent security domains — identity, network, data, workload and configuration state. An edge that only one domain supports does not survive.

> **Why this gate exists.** An unverified path is worse than no path: it spends a security team's credibility on a route that may not exist. Verification is the difference between a graph that suggests and a graph that asserts.

06 MITRE mapping, choke points, and price

Each verified hop maps to a MITRE ATT&CK technique, so the route reads as an attack narrative rather than a list of resource IDs. The full set of paths then reduces to its **top five choke points** — the nodes appearing on the most confirmed routes, where one change removes the most attack surface.

Finally the exposure is priced with FAIR, producing an Annualised Loss Expectancy per path and a roll-up across the estate. Method detail is in Whitepaper 02.

07 Summary

One claim, repeated at every stage: a finding is a fact, and only a verified route is a decision.

- Severity ranks findings against each other; a graph ranks routes against reality.
- 7 clouds, read-only and agentless, into one property graph with typed abuse edges.
- Breadth-first search from reachable entry points to designated crown jewels.
- Five-domain verification before any path is called confirmed.
- MITRE mapping for narrative, choke points for leverage, FAIR for the number.
- Onam does not block at runtime. It tells you, reproducibly, which few fixes matter most.

ONAM SECURITY

TECHNICAL WHITEPAPER · 02

Cloud risk in dollars

How Onam prices a verified attack path with FAIR — using named, external inputs a board can defend.

FAIR / ALE | IBM 2024 costs | Sensitivity × | Regulatory × | Exposure roll-up

THE ARGUMENT, IN ORDER

00 Executive summary

Severity tells you what is broken. Dollars tell you what to fix first.

Security teams are drowning in "criticals". Boards are not asking how many — they are asking how much. This paper explains how Onam turns a verified attack path into a defensible financial figure using FAIR, the open standard for risk quantification, so the conversation moves from a spreadsheet of severities to a single number a board can act on.

FAIR — Factor Analysis of Information Risk — is a published, peer-reviewed methodology maintained as an open standard by The Open Group. It decomposes risk into estimable factors and produces an Annualised Loss Expectancy (ALE) in currency. Onam implements FAIR as the fourth layer of its pipeline: after a path is found, verified and mapped to MITRE ATT&CK, it is priced. The pricing is not a marketing gauge invented in-house; it is a recognised model fed with named, external inputs.

> **Why this is the wedge.** A security graph and attack paths are now common across the CNAPP market. A defensible dollar figure — derived from a published model and named external cost data, attached to a specific verified path — is not. It is the difference between "we have 200 criticals" and "this one over-privileged role carries the largest single slice of our quantified exposure."

01 The FAIR model, briefly

The model is not ours. FAIR is an open, published standard, which is exactly why the number survives scrutiny.

FAIR expresses risk as the product of how often a loss happens and how large it is when it does.

The strength of FAIR is that each factor is estimable and inspectable. When Onam prices a path, a reviewer can open the calculation and see which input drove the number — not a black-box score between 0 and 100.

02 The inputs — named, external, defensible

The magnitude inputs come from a published benchmark, not from us.

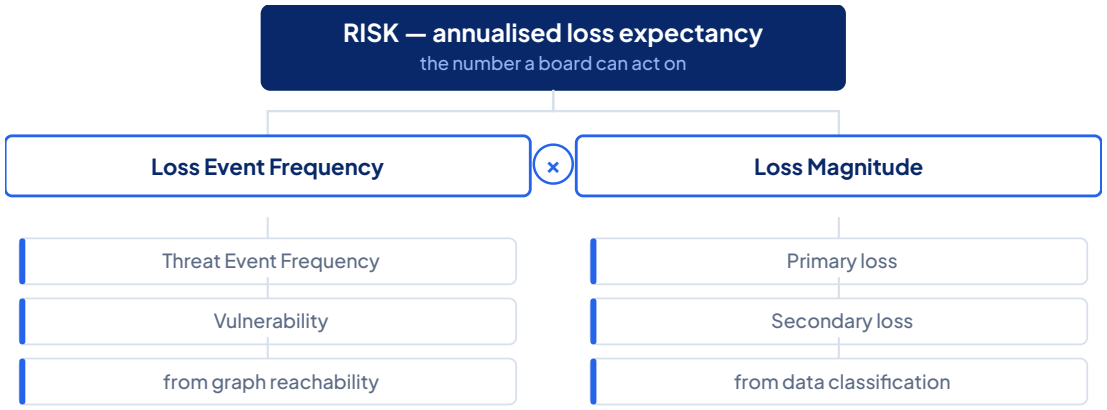

FIGURE 1 — FAIR DECOMPOSITION

Every leaf is computed, not estimated. That is the whole difference between a priced exposure and a workshop number.

A dollar figure is only as credible as the numbers behind it. Onam deliberately sources its magnitude inputs from a recognised external benchmark and applies transparent multipliers.

Input	Source / basis	Effect
Per-record breach cost	IBM Cost of a Data Breach 2024, by industry	Healthcare ≈ $10.93/record; default ≈ $4.45/record
Data-sensitivity multiplier	Onam data classification (DSPM)	Restricted ×3.0 → Confidential → Internal → Public
Regulatory multiplier	Applicable regime (GDPR, HIPAA, PCI)	Raises magnitude where statutory penalty applies
Frequency & vulnerability	Graph-derived reachability and control state	Replaces a workshop estimate with evidence

Every multiplier is visible and adjustable. A customer who disagrees with an input can change it and see the figure move — which is the point of an inspectable model.

> **The honest limit.** A quantified figure is an estimate, not an invoice. FAIR produces a defensible range built on stated assumptions; it does not predict what a specific breach will cost. Anyone presenting it as a guarantee is misusing the model, including us.

03 From path to price

Pricing happens after verification, never before. An unverified path is not priced, because a number attached to a route that does not exist is worse than no number.

1. **Find.** Graph traversal from an entry point to a crown jewel across 7 clouds.
2. **Verify.** Five-domain verification before a path is called confirmed.
3. **Map.** MITRE ATT&CK technique mapping for the steps on the path.
4. **Price.** FAIR applied to the verified path, using the inputs above.

04 Rolling up — from one path to a portfolio

A remediation queue you can argue with is worth more than one you have to trust.

Individual path prices aggregate into an exposure roll-up: total quantified exposure, the largest contributors, and the choke points where a single fix removes the most dollars. That is the view a board can act on, and the view that makes a remediation queue arguable rather than arbitrary.

Because the same graph and the same 9,853 CSPM posture rules feed both the compliance and the risk lenses, a control gap and a priced path are two readings of one underlying fact — not two systems to reconcile.

05 Summary

Every number in this paper can be taken apart — that is the point of it.

- FAIR is an open, published standard — the model is not ours, only the implementation is.
- Magnitude inputs are named and external (IBM Cost of a Data Breach 2024), not invented.
- Every factor is inspectable; a reviewer can see which input drove the number.
- Pricing follows verification. An unverified path is never priced.
- The output is an estimate with stated assumptions, not a prediction of a specific loss.

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.

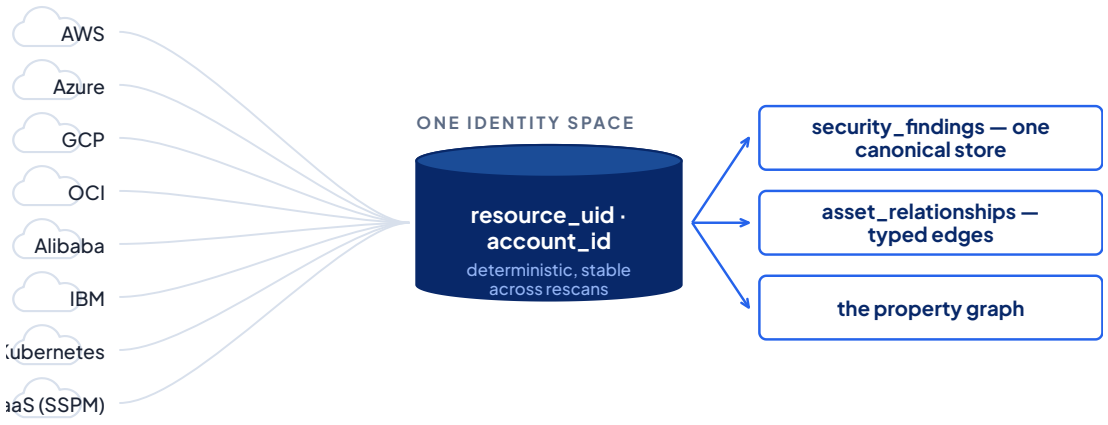

FIGURE 1 — MANY CLOUDS, ONE NORMALISED STORE

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.

ONAM SECURITY

TECHNICAL WHITEPAPER · 04

Security & trust

How Onam connects, what it stores, what it never stores, and how tenants stay isolated — the trust case for a read-only platform.

Least privilege

No stored secrets

Metadata not data

Per-tenant isolation

SOC 2 · ISO · PCI

THE ARGUMENT, IN ORDER

00 Executive summary

A security tool has to earn more trust than it asks for.

Onam reads your entire cloud. That access is exactly why our own security posture has to be exemplary and legible. This paper documents how Onam connects, what it stores, what it deliberately never stores, and how tenants are isolated — so a security team can grant access with confidence rather than hope.

The design principle is minimality: take the least privilege that permits a security read, hold no long-lived secrets, and retain only the metadata needed to describe a risk — never the data itself. Where a control is a stated certification, it is named; where something is planned rather than achieved, this paper does not claim it.

> **The short version.** Read-only, agentless, short-lived tokens, per-tenant isolation with per-tenant keys, and a retention policy that keeps labels and locations but not contents. The rest of this paper is the detail behind each of those words.

01 Least-privilege connection

No long-lived cloud credential is ever stored — only a reference, exchanged for a token that expires within the hour.

Every cloud is connected with the lowest standing privilege that still allows a security read, and no long-lived cloud credential is ever stored — only a reference (a role ARN or principal ID) held in a secrets manager and encrypted with a managed key. At scan time that reference is exchanged for a short-lived token.

Cloud	Mechanism	Privilege
AWS	Cross-account role, unique External ID per tenant	SecurityAudit + ReadOnlyAccess; STS ≤ 60 min
Azure	Entra service principal	Reader + read-only Graph
GCP	Service account or keyless WIF	6 read-only roles
OCI	IAM user + signing key	inspect + read only
Alibaba / IBM	RAM read-only · Service ID	Read-only; key exchanged for short-lived Bearer
Kubernetes	Read-only ServiceAccount	get / list / watch

> **Confused-deputy defence.** The AWS connection requires a unique External ID per tenant, so one tenant's role can never be assumed on behalf of another. Tokens expire in an hour or less, and secret material is never written to logs.

02 What Onam never stores

The most important trust control is what does not enter the system.

Onam keeps the metadata required to describe and price a risk, and deliberately discards everything else.

This is what makes "we read your entire cloud" a bounded statement. Onam knows that a bucket holds restricted data and where it sits on a path; it does not hold the objects in it.

03 Tenant isolation

Isolation here is separate keys and separate execution, not a tenant column in a shared table.

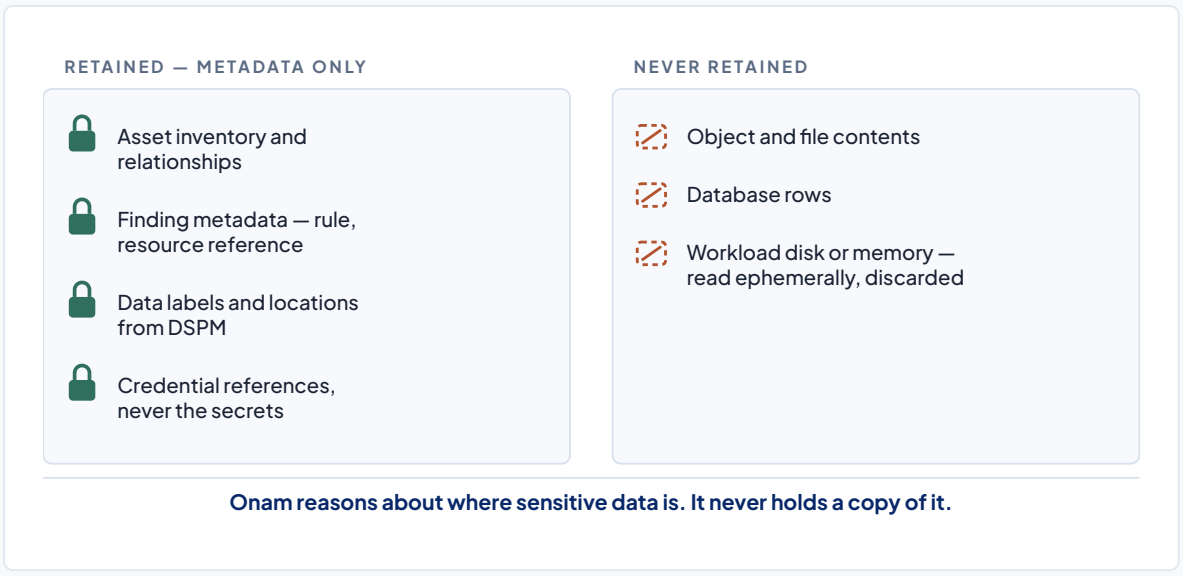

FIGURE 1 — RETAINED VS NEVER RETAINED

The right column is the commitment. Everything in it is read, used, and discarded in the same pass.

Each tenant's data is isolated with per-tenant keys, so a defect in one tenancy cannot expose another. Scan execution, storage and key material are separated per tenant rather than partitioned by a shared identifier in a shared store.

04 Agentless by design — and what that costs

Nothing is deployed on your workloads. There is no kernel module, no sidecar, no daemon to patch, and no agent fleet to roll out or roll back. Onboarding is a role grant, and removing Onam is revoking it.

> **The honest limit.** Agentless is a trade, not a free win. Onam has no live inline runtime enforcement: it evaluates posture continuously and detects from audit logs, but it cannot block a process on a host the way an agented product can. If inline blocking is a requirement, that requirement is not met by this architecture.

05 Certifications and claims discipline

Nothing on a roadmap is written in the present tense in this paper.

Where Onam holds a certification, this paper names it. Where a control is implemented but not certified, it is described as implemented. Where something is on a roadmap, it is labelled planned and never stated in the present tense.

That discipline is itself a security control: a trust document that overstates is a trust document that will be checked, disbelieved, and then discounted entirely.

06 Summary

Every claim in this paper is either a control you can inspect or a limit we state.

- Least privilege per cloud, with no long-lived credential stored — only a reference, exchanged for a token of an hour or less.
- Metadata is retained; contents are not. Labels and locations, never rows and objects.
- Per-tenant isolation with per-tenant keys.
- Agentless by design, and honest that this means no inline runtime enforcement.
- Present tense means shipped. Planned means planned.

ONAM SECURITY

TECHNICAL WHITEPAPER · 05

Compliance, mapped once

How Onam evaluates a control once and reports it against 78 frameworks — and connects every gap to a priced attack path.

One rule → many controls

Rules as code

78 frameworks

Custom frameworks

Gap → priced path

THE ARGUMENT, IN ORDER

00 Executive summary

Evaluate the control once. Report it against every framework.

Compliance work is mostly duplication: the same control, re-checked and re-evidenced for PCI, then SOC 2, then ISO, then a customer questionnaire. Onam removes the duplication by decoupling the check from the framework. Rules are evaluated once as code; a single finding maps to many controls across 78 frameworks at once.

This paper explains the mapping model, why rules-as-code makes framework coverage extend automatically, and how compliance and attack-path reasoning reinforce each other — so a control gap is not just a failed check but a node on a priced path.

> **An honest framing.** Framework count is a coverage feature, not a differentiator on its own — several platforms publish large numbers. Onam's distinct value is the architecture that produces the mapping (one rule → many controls, auto-extending) and the ability to connect a compliance gap to a priced attack path. This paper leads with the mechanism, not the number.

01 One finding, many controls

Frameworks disagree about wording far more than they disagree about controls.

A framework is, at bottom, a set of controls. Many controls across many frameworks ask for the same underlying thing — "encrypt data at rest", "restrict public access", "enforce MFA". Onam evaluates that underlying condition once and maps the result to every control it satisfies or violates.

02 Rules as code — why coverage extends itself

A new rule extends every framework it maps to, and a new framework inherits every rule already written.

Onam's rules are versioned YAML the platform loads and evaluates; nothing is hardcoded. This is what makes framework coverage compound rather than accumulate by hand. When a new rule is added, it automatically extends every framework whose controls it maps to. When a new framework is onboarded, its controls map to existing rule IDs — no new checks to write.

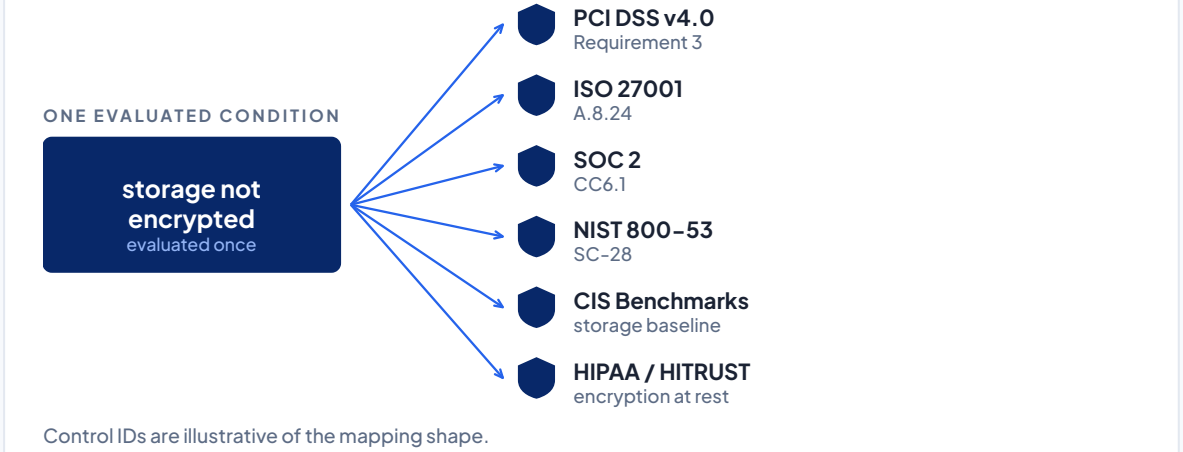

FIGURE 1 — THE MAPPING FAN-OUT

The duplication is in the reporting, not the work. Control IDs are illustrative of the mapping shape.

1. **Add a rule.** A new YAML rule ships; every mapped framework gains coverage for the condition it checks.
2. **Add a framework.** A new framework's controls map to existing rule IDs; coverage appears without new detection code.
3. **Add a custom framework.** A Rule Builder and custom-framework mapping let a team model an internal standard against the same rule library.

> **The compounding effect.** Because checks and frameworks are decoupled, work done for one obligation pays down others automatically. A rule written to satisfy an internal standard may also close a PCI and an ISO control the same day — without anyone re-authoring the check.

03 Frameworks supported

Coverage here means continuous evaluation, not an audit export generated the week before.

Onam supports 78 frameworks out of the box across regulatory, industry and best-practice standards — for example PCI DSS v4.0, SOC 2, ISO 27001/27017, HIPAA/HITRUST, GDPR, NIST 800-53 and CIS benchmarks — with custom frameworks mappable to rule IDs. Coverage is delivered as continuous evaluation, not a point-in-time audit export.

The mapping rests on a catalog of 11,433 rule definitions across 7 clouds and 549 cloud services, of which 9,853 are CSPM posture rules.

04 Compliance meets the attack path

Two control failures of equal audit weight are not equal risk.

A failed control is a compliance fact. The same failing condition is also a node in the property graph — which means Onam can tell you not just that a control failed, but whether its failure lies on a verified, priced route to a crown jewel. That reframes remediation: two "failures" of equal audit weight are not equal if one sits on a choke point and the other reaches nothing.

Lens	Question it answers	What it ranks by
Compliance	Which controls are failing, and where is the evidence?	Framework obligation
Risk	Which failure is on a reachable path to something that matters?	Priced exposure (FAIR/ALE)

Read together, the two lenses turn an audit backlog into an ordered queue: the control gaps that are also choke points come first, because fixing one of them closes an obligation *and* removes a route.

05 What framework count does and does not prove

A large framework number proves breadth of mapping. It does not prove depth of evaluation, quality of evidence, or that the controls most relevant to your obligations are covered well.

- **It does prove** that a platform has invested in mapping, and that onboarding a new obligation is unlikely to require new detection work.
- **It does not prove** that any individual control is checked the way your auditor expects, nor that evidence export matches your reporting format.
- **The question to ask any vendor**, including us: show me the rule behind this control, and the evidence it produces.

> **The honest limit.** Onam evaluates posture continuously and maps it to controls. It does not replace an auditor, and it does not produce a signed attestation. What it produces is the evidence an auditor asks for, kept current between audits rather than assembled the week before one.

06 Summary

Fix the condition once; every framework that maps to it updates itself.

- One evaluated condition maps to many controls across 78 frameworks — fix once, update everywhere.
- Rules-as-code means coverage extends automatically when either a rule or a framework is added.
- A control gap that sits on a priced attack path is not the same as one that reaches nothing, and Onam ranks accordingly.
- Framework count is a coverage feature. The architecture that produces the mapping is the differentiator.