SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Designing for Convergence: Architectural Patterns for Unifying Independently Built Security Data Platforms Across Organizational Boundaries

Rahul Kizhakkepachilamakol

Amazon Web Services, USA. ORCID ID: https://orcid.org/0009-0008-4437-0325

Abstract

Cloud-scale security organizations routinely accumulate independently built data platforms for vulnerability management, application security, compliance, and threat detection, each optimized for a specific team's mandate but collectively unable to provide the cross-domain visibility that modern defense requires. This fragmentation follows directly from Conway's Law: system boundaries mirror the organizational boundaries that produced them. This article develops an architectural framework for converging such platforms without disruptive replacement, and argues that a canonical data model becomes practically maintainable only when paired with machine learning-based entity resolution and schema matching, which turn cross-domain data reconciliation from a brittle manual exercise into a scalable, continuously adapting process. Drawing on domain-driven design, data mesh architecture, and the software engineering literature on organizational alignment, the framework positions convergence along a spectrum from loose federation to full unification and specifies four architectural mechanisms a canonical schema layer, machine learning-based entity and schema matching, API-first domain gateways, and progressive migration under federated governance that organizations can combine according to their technical debt and readiness for change. An illustrative application to the convergence of vulnerability management, application-security, and compliance data platforms shows how these mechanisms interact in practice. The analysis also identifies the limits of machine learning-based matching in security-critical contexts, where a false merge of unrelated findings carries operational risk that differs from typical data-integration use cases. The framework contributes a synthesis that existing literature, which treats organizational design, data architecture, and automated entity matching as separate concerns, has not yet provided for security data specifically.

Keywords: platform convergence; security data platform; data mesh; domain-driven design; entity resolution; schema matching; artificial intelligence; machine learning

1. Introduction

Security organizations operating at cloud scale confront a paradox: the same specialization that lets a team build an effective vulnerability scanner, application-security analyzer, or compliance monitor also all but guarantees that the platform will not talk to its neighbors. Each team optimizes for its own mandate, its own service-level objectives, and its own consumers, and each produces a data platform that reflects that mandate rather than the cross-domain view a defender actually needs. A vulnerability finding sitting in one system cannot be automatically linked to the application-security weakness that makes it exploitable, or to the compliance obligation that determines how quickly it must be remediated, unless someone builds that connection by hand. Melvin Conway observed nearly six decades ago that any organization that designs a system will produce a design whose structure mirrors the organization's own communication structure (Conway, 1968). In cloud security specifically, that observation has become close to a law of nature: when a vulnerability management team, an application-security team, and a compliance team sit under different reporting lines with different budgets and different roadmaps, their data platforms diverge in exactly the ways their organization charts diverge, independent of whether that divergence serves the defender.

The consequences are not abstract. Security teams that must query several platforms to reconstruct the full context of a single finding lose time that attackers do not. A 2020 industry survey of security professionals found that respondents managed more than 45 different security tools on average, and that responding to a single incident typically required coordinating information across around 19 of them (Ponemon Institute, 2020). Every additional platform a team must consult to answer a routine question is a tax on the speed of that

team's decisions, and in security operations decision speed is close to the entire product. The costs compound in less visible ways as well. Infrastructure is duplicated across platforms that ingest overlapping data, and engineering capacity goes toward point-to-point connectors rather than detection capability. The same underlying entity an asset, a vulnerability, an identity ends up represented differently in each system, with no reliable mechanism for confirming that three separately recorded findings actually describe one thing.

This article treats the convergence of independently built security data platforms as an architectural problem with an organizational cause, and argues that the mechanism most often missing from proposed solutions is not organizational will but a scalable way to reconcile heterogeneous data automatically. Two literatures already speak to parts of this problem without fully joining them. The software architecture literature on domain-driven design and data mesh architecture offers a vocabulary for how domain-owned platforms can interoperate without being collapsed into a single monolith (Evans, 2003; Dehghani, 2022). The data-management literature on entity resolution and schema matching offers machine learning techniques that can recognize when two differently structured records refer to the same real-world entity, replacing what would otherwise be a manual mapping exercise that grows combinatorially with the number of platforms involved (Binette & Steorts, 2022; Huang et al., 2023). Neither literature, taken alone, addresses what changes when the entities being matched are security findings, where an incorrect match treating two unrelated vulnerabilities as one, or conflating two distinct assets creates operational risk of a different character than a duplicate customer record in a retail database.

The framework developed here integrates these two literatures around a specific claim: a canonical data model is the architectural centerpiece of convergence, but a canonical model only stays maintainable across platforms that were never designed to share one when it is paired with machine learning-based entity resolution and schema matching. Hand-written mapping rules, by contrast, must be re-engineered every time a source platform's schema drifts. Artificial intelligence and machine learning techniques for entity and schema matching are therefore treated in this article not as an optional enhancement to convergence but as the mechanism that determines whether convergence remains a one-time integration project or becomes a durable architectural property of the organization's security data.

The contribution of this article is a synthesis. It connects an organizational-design account of why security platforms fragment, an architectural account of how domain-owned data platforms can interoperate, and a data-management account of how machine learning makes cross-domain reconciliation tractable, into a single framework aimed specifically at security data. Existing treatments of platform convergence tend to address one of these three concerns at a time and treat the others as implementation detail; this article treats their interaction as the central design problem, and grounds each proposed mechanism in a review of the corresponding literature rather than in architectural intuition alone.

The remainder of the article proceeds as follows. Section 2 reviews the literatures on organizational alignment, data mesh and domain-driven design, and machine learning-based entity resolution that inform the framework, and identifies the specific gap it addresses. Section 3 describes the analytical approach used to derive and organize the framework. Section 4 presents four architectural mechanisms a canonical schema layer, machine learning-based entity and schema matching, API-first domain gateways, and progressive migration under federated governance positioned along a spectrum from loose federation to full unification. Section 5 applies the framework to an illustrative convergence scenario spanning vulnerability management, application security, and compliance monitoring. Section 6 discusses the framework's implications and the specific limitations of machine learning-based matching in security-critical settings, Section 7 states the limitations of the analysis, and Section 8 concludes.

Table 1. The Integration Tax: Documented Costs of Security Platform Fragmentation

Cost Category	Description	Source
Tool volume	Organizations report managing more than 45 different security tools on average, with a single incident typically requiring coordination across around 19 of them.	Ponemon Institute (2020)
Infrastructure duplication	Independently built platforms ingest and store overlapping data, duplicating infrastructure that a shared canonical layer could consolidate.	Putrama & Martinek (2024)

Integration engineering effort	Point-to-point connectors between platforms require ongoing engineering maintenance that grows with the number of platform pairs involved.	Ruíz-Ceniceros et al. (2023)
Entity ambiguity	The same asset, vulnerability, or identity is represented with different identifiers across platforms, with no reliable mechanism to confirm they describe one entity absent an entity resolution layer.	Binette & Steorts (2022)
Correlation blindness	Manually authored correlation rules in SIEM and threat-intelligence platforms struggle to keep pace with continuously evolving data sources.	González-Granadillo et al. (2021); Zang et al. (2023)

2. Literature Review

2.1 The Organizational Roots of Platform Fragmentation

Conway's (1968) observation that a system's structure mirrors the communication structure of the organization that built it has held up across five decades of software engineering practice, and cloud security organizations illustrate it with particular force. When a vulnerability management team, an application-security team, and a compliance team operate under separate leadership, separate budgets, and separate roadmaps, each will build a data platform optimized for its own domain rather than for the composite view that a security analyst eventually needs. The resulting landscape is not badly designed in any local sense; each individual platform may reflect sound engineering judgment for its narrow purpose. The fragmentation appears only in aggregate, once an analyst or an automated system needs information that spans more than one platform's boundary.

The organizational-design response to Conway's Law is what Skelton and Pais (2019) term the inverse Conway maneuver: deliberately shaping team boundaries so that the architecture an organization wants emerges from the communication structure it establishes, rather than attempting to correct architecture after the fact. Skelton and Pais's Team Topologies framework distinguishes stream-aligned teams, platform teams, and enabling teams, and argues that convergence initiatives fail more often for organizational reasons unclear ownership, absent platform-team capacity, unresolved cognitive-load conflicts than for purely technical ones. This suggests that any architectural framework for security-platform convergence has to specify not only the target architecture but also the team structure and governance model capable of sustaining it, a point returned to in Section 4.5.

Cloud security organizations add a further complication that the general Conway's Law literature does not address directly. The teams producing independent platforms are frequently organized around threat or asset domains vulnerabilities, application code, cloud configuration, identity that were themselves drawn along historical, often acquisition-driven or compliance-driven lines rather than along lines optimized for downstream data interoperability. A platform boundary that made organizational sense when a domain team was formed does not necessarily correspond to a boundary that makes sense for the queries an analyst or an automated system needs to run years later, once the organization's security posture depends on correlating findings the original team boundaries never anticipated needing to correlate. This gap between the organizational logic that produced a platform boundary and the analytical logic that governs how findings should ideally be queried is, in effect, the problem this article's framework exists to address.

2.2 Domain-Driven Design and Data Mesh as Architectural Responses

Domain-driven design, as formulated by Evans (2003), provides the conceptual vocabulary most directly relevant to convergence without consolidation: bounded contexts, in which a domain team owns its own model and vocabulary, and context maps, which specify how different bounded contexts translate concepts across their boundaries. A 2024 evidence-based investigation of domain-driven design applied to microservice systems found that teams using bounded-context decomposition reported clearer service boundaries and fewer unintended cross-service dependencies than teams that decomposed services along purely technical lines. The same investigation also noted that maintaining context maps as systems evolve requires ongoing governance rather than a one-time design exercise (Zhong et al., 2024). This empirical caveat matters for

convergence specifically: a context map that is accurate at design time and unmaintained thereafter degrades into exactly the kind of undocumented, drifting integration that convergence is meant to eliminate.

Data mesh architecture extends this domain-oriented thinking from services to data specifically. Dehghani (2022) proposes four principles domain-oriented decentralized data ownership, data treated as a product, self-serve data infrastructure, and federated computational governance that together aim to let domain teams retain ownership of their data while making that data interoperable at organizational scale. Machado, Costa, and Santos (2021) formalize these principles into an architectural description, emphasizing that the approach substitutes federated standards for centralized control rather than eliminating standards altogether. A systematic review of data mesh gray literature, drawing on 114 industry sources, found that reported implementations most commonly struggled with establishing the shared infrastructure and governance layer rather than with the conceptual principles themselves, suggesting a gap between data mesh as a design philosophy and data mesh as a buildable platform (Goedegebuure et al., 2024). API-first design complements both domain-driven design and data mesh by treating the interface contract, not the internal implementation, as the primary object of standardization (Fielding, 2000). Empirical work on microservice API evolution finds that teams frequently underestimate the coordination cost of changing a published API once external consumers depend on it, which argues for treating API contracts in a convergence architecture as comparably durable commitments to the canonical schema itself (Lercher et al., 2024). Migration research more broadly confirms that organizations moving toward domain-decomposed, service-oriented architectures encounter as many organizational and process challenges as technical ones, reinforcing the point raised in Section 2.1 (Ayas, Leitner, & Hebig, 2023).

2.3 Machine Learning-Based Entity Resolution and Schema Matching

Entity resolution determining whether two or more records refer to the same real-world entity and schema matching determining how fields in one data source correspond to fields in another are the data-management problems that arise the moment two independently built platforms need to share information. Binette and Steorts (2022) characterize the modern entity resolution pipeline as a sequence of blocking (reducing the number of candidate pairs to compare), pairwise or set-based matching, and clustering, and note that the matching stage has shifted decisively from hand-tuned similarity rules toward learned representations. Deep learning approaches to entity matching now commonly rely on active learning, in which a model iteratively selects the most informative unlabeled record pairs for human review, which reduces the labeling burden that previously made supervised matching impractical for organizations without large curated training sets (Huang et al., 2023). A hybrid active-learning framework applied to sparse, real-world entity resolution datasets reported that combining active learning with auxiliary weak-supervision signals produced usable matching models with meaningfully less labeled data than fully supervised baselines required, a property directly relevant to security data platforms, where cross-domain labeled examples of matching findings are scarce (Jabrane et al., 2024).

Schema matching the companion problem of aligning field-level semantics rather than record-level identity has followed a similar trajectory. A 2024 architectural proposal for a dynamic canonical data model demonstrates that static, hand-authored mapping rules become a maintenance liability once more than a handful of source systems must be reconciled, and argues for architectures in which the mapping layer can be updated as source schemas evolve rather than rebuilt (Ruíz-Ceniceros et al., 2023). A broader review of heterogeneous data integration challenges reaches a similar conclusion from the data-engineering side, observing that schema heterogeneity, rather than sheer data volume, is the more persistent obstacle to integration in organizations that accumulate systems incrementally over time (Putrama & Martinek, 2024). Taken together, this literature supports treating machine learning-based entity and schema matching as a maintenance-reducing mechanism rather than a one-time integration accelerator: its principal advantage over rule-based mapping is that it can be retrained as sources drift, rather than requiring an engineer to notice the drift and rewrite the rule.

2.4 Security-Specific Data Integration Literature

Within the security domain specifically, integration research has concentrated on security information and event management (SIEM) platforms and on threat-intelligence fusion. A widely cited analysis of SIEM deployment in critical infrastructure environments describes correlation and normalization across heterogeneous log and alert sources as the central technical challenge of SIEM operation, and observes that most production deployments still rely substantially on manually authored correlation rules rather than learned models (González-Granadillo, González-Zarzosa, & Diaz, 2021). Threat-intelligence fusion research has moved further toward automation: a framework for reconstructing attack scenarios from heterogeneous threat-intelligence sources demonstrates that fusing indicators across sources can reveal attack sequences

invisible to any single source, but the framework depends on a substantial preprocessing stage to reconcile inconsistent entity and indicator representations before fusion becomes possible (Zang et al., 2023). A 2025 systematic review of cyber threat intelligence practice, synthesizing 43 peer-reviewed studies, concludes that data standardization across sources and organizations remains one of the most persistent open challenges in the field, ahead of concerns about analytic sophistication (Santos et al., 2025). On the vulnerability side, recent work applying machine learning to CVSS-based vulnerability prioritization shows that models incorporating environmental and contextual information from multiple data sources materially outperform static severity scores alone for prioritization purposes, which depends in practice on those sources first being reconciled into a comparable representation (Balsam et al., 2025).

2.5 Gap in the Literature

Each of these literatures addresses a necessary piece of security-platform convergence without addressing the whole. The organizational-design literature explains why platforms fragment and what team structures can prevent further fragmentation, but treats the technical mechanism of reconciling already-fragmented data as outside its scope. The domain-driven design and data mesh literature specifies how domain-owned platforms should be organized and governed, but is largely agnostic to the underlying data-matching problem that determines whether a canonical model is achievable in practice. The entity resolution and schema matching literature supplies increasingly capable machine learning techniques for exactly that data-matching problem, but is written for general-purpose data integration and does not address the specific risk profile of security data, where an incorrect automated match can suppress a genuine finding or merge two unrelated ones. The security-specific integration literature, in turn, documents the operational consequences of unreconciled data but has not yet drawn systematically on the entity resolution literature's more recent machine learning methods. This article addresses that combined gap by proposing a framework in which organizational design, canonical-model architecture, and machine learning-based matching are treated as interdependent components of a single convergence strategy for security data platforms specifically.

3. Analytical Approach

This article makes a conceptual and architectural contribution rather than an empirical one, and the approach used to derive it should be read in that light. The framework was developed through a structured synthesis of three literatures organizational design and Conway's Law, domain-driven design and data mesh architecture, and machine learning-based entity resolution and schema matching combined with the interpretive lens of the author's professional experience as a practitioner in cloud-scale security data-platform engineering. No original organizational data, benchmark, or controlled comparison is reported here; where the article draws on quantitative findings, those figures are drawn exclusively from the cited peer-reviewed or industry-survey sources listed in the references, and are reported with the qualifiers those sources used.

Literature was identified through searches of IEEE Xplore, the ACM Digital Library, ScienceDirect, SpringerLink, and Google Scholar. Search terms combined “data mesh,” “domain-driven design,” “entity resolution,” “schema matching,” “Conway's Law,” “microservice migration,” “SIEM,” and “threat intelligence fusion.” Sources were retained for inclusion if they were peer-reviewed, directly relevant to at least one of the three literatures described above, and independently verifiable through the publisher's own record. For claims involving specific quantitative findings, retained sources were restricted to those published in 2020 or later, with the exception of foundational theoretical works cited for their conceptual contribution rather than for a numerical claim.

The framework itself was organized by identifying, for each of the three literatures, the architectural mechanisms it proposes or implies. Those mechanisms were then examined for places where two literatures address the same underlying design decision for example, where data mesh's principle of federated governance and Team Topologies' account of platform-team responsibility both bear on how a convergence initiative should be staffed and sequenced. Mechanisms that appeared, explicitly or implicitly, in at least two of the three literatures were retained as core components of the framework. Mechanisms specific to only one literature were retained only where the security-specific integration literature reviewed in Section 2.4 independently corroborated their relevance to security data platforms.

The framework is illustrated in Section 5 through a walkthrough scenario rather than a controlled empirical evaluation. The scenario is constructed to be representative of the kind of convergence problem documented in the security-specific integration literature unifying vulnerability management, application-security, and compliance-monitoring data. It is a composite illustration rather than a report of a specific organization's deployment, and no proprietary system names, internal metrics, or unpublished data from any specific

organization are used in its construction. This approach follows established precedent in the software architecture literature, where conceptual frameworks for data mesh and domain-driven design were themselves introduced and refined through illustrative scenarios and practitioner synthesis, prior to and alongside subsequent empirical investigation of their adoption (Evans, 2003; Dehghani, 2022; Ayas, Leitner, & Hebig, 2023). The limitations that follow from this approach, particularly the absence of a controlled comparison against alternative convergence strategies, are discussed explicitly in Section 7.

4. A Framework for Platform Convergence

The framework proposed here treats convergence not as a single destination but as a spectrum of architectural commitments, and specifies four mechanisms that organizations combine differently depending on where along that spectrum they intend to sit. Table 2 summarizes the spectrum; the four subsections that follow describe the mechanisms in turn.

4.1 The Convergence Spectrum

At one end of the spectrum, federation preserves each platform's autonomy while establishing minimal interoperability contracts: platforms continue to operate independently but expose standardized query interfaces that allow a consumer to retrieve data without understanding each platform's internal representation. At the other end, full unification consolidates platforms into a single system with one data model, shared infrastructure, and common operational practice. Between these extremes, two intermediate patterns are most consistent with the literature reviewed in Section 2. A canonical-layer pattern introduces a shared data model and a translation layer above existing platforms, enabling cross-domain correlation without requiring any platform to change its internal representation. A domain-gateway pattern establishes API boundaries that abstract platform-specific detail behind stable interfaces, so that platforms can be consolidated or replaced behind the gateway without disrupting consumers (Fielding, 2000). Which pattern an organization should adopt for a given pair of platforms depends on how much functional overlap exists between them and how much organizational appetite exists for consolidation rather than interoperation a distinction the data mesh and domain-driven design literatures treat as a governance decision rather than a purely technical one (Dehghani, 2022; Evans, 2003).

Table 2. Comparative Overview of Convergence Patterns Along the Federation-to-Unification Spectrum

Pattern	Platform Autonomy	Primary Mechanism	Organizational Disruption	Best Suited For
Federation	High platforms remain fully independent	Standardized query interfaces only	Low	Platforms with minimal functional overlap needing occasional cross-referencing
Canonical layer	High platforms unchanged internally	Shared canonical data model plus machine learning-based entity and schema matching	Low to moderate	Platforms with substantial overlapping entities but distinct finding types
Domain gateway	High platforms abstracted behind a stable API	API-first gateway routing queries across platforms	Moderate	Platforms with complementary, non-overlapping functions that must be queried together
Full unification	Low platforms consolidated	Single shared infrastructure and data model	High	Platforms with near-total functional overlap and low ongoing need for domain specialization

4.2 The Canonical Data Model as Convergence Foundation

The most consequential architectural decision in any convergence effort is the establishment of a canonical data model: a standardized representation of security findings that participating platforms agree to produce and consume, whether directly or through a translation layer. Without a shared model, cross-platform correlation requires ad hoc, pairwise semantic reconciliation between every pair of platforms, an approach whose engineering cost grows with the square of the number of platforms involved. With a canonical model, each platform needs only a single mapping to and from the shared representation, which converts an integration problem that scales quadratically into one that scales linearly. Multi-vendor efforts to standardize security-event schemas across the industry reflect the same underlying logic at a cross-organizational scale, though adopting an external schema standard is only the starting point: organizations still need domain-specific extensions, mapping rules for legacy formats, and a validation pipeline that enforces conformance over time (Putrama & Martinek, 2024; Ruíz-Ceniceros et al., 2023).

The canonical model's practical difficulty is not in its initial design but in keeping the mapping between it and each source platform accurate as those platforms evolve. A static, hand-authored mapping degrades as soon as a source platform adds a field, renames an attribute, or changes how it represents severity, and detecting that drift typically depends on an engineer noticing a downstream anomaly rather than on any property of the mapping itself. This is precisely the maintenance burden that Section 4.3 argues machine learning-based matching is positioned to reduce.

4.3 Machine Learning-Based Entity Resolution and Schema Matching as the Convergence Enabler

If the canonical data model is the architectural centerpiece of convergence, machine learning-based entity resolution and schema matching are what keep that centerpiece maintainable once more than a small number of platforms participate. Two matching problems recur throughout convergence work. The first is entity resolution across platforms: determining that a vulnerability record in one platform, an application-security finding in a second, and a compliance exception in a third all pertain to the same underlying asset or the same underlying weakness, even though each platform assigns its own identifiers and records different attributes. The second is schema matching within the canonical model itself: determining how a newly onboarded platform's fields correspond to the canonical schema's fields, particularly when field names, units, and severity scales differ across platforms that were never designed with each other in mind.

For both problems, the literature reviewed in Section 2.3 supports treating machine learning-based matching as materially more sustainable than rule-based mapping at the scale cloud security organizations operate. Active-learning approaches to entity matching reduce the volume of manually labeled examples required to train a usable matching model, which matters directly for security data, where an organization is unlikely to have large curated sets of confirmed cross-platform matches to draw on (Huang et al., 2023; Jabrane et al., 2024). A learned matching model can also be retrained as source platforms drift, whereas a hand-written mapping rule has to be rewritten by an engineer who first has to notice that it broke. This distinction is what separates convergence as a one-time integration project from convergence as a durable property of the architecture. A canonical model maintained by static rules re-accumulates the same brittleness that motivated convergence in the first place. A canonical model maintained by a retrainable matching layer degrades more gracefully as the underlying platforms continue to change.

This mechanism is not without risk, and the risk is specific to security data in a way the general entity-resolution literature does not fully address. A false match treating two distinct vulnerabilities as the same finding, or merging records for two different assets can suppress a genuine finding from an analyst's view or misattribute remediation responsibility, in a way that a duplicate record in a customer database typically does not. Section 6.2 returns to this limitation directly; the framework's position is that machine learning-based matching should be deployed with confidence thresholds and human review paths calibrated to this asymmetry, rather than adopted as a fully automated replacement for analyst judgment on ambiguous matches.

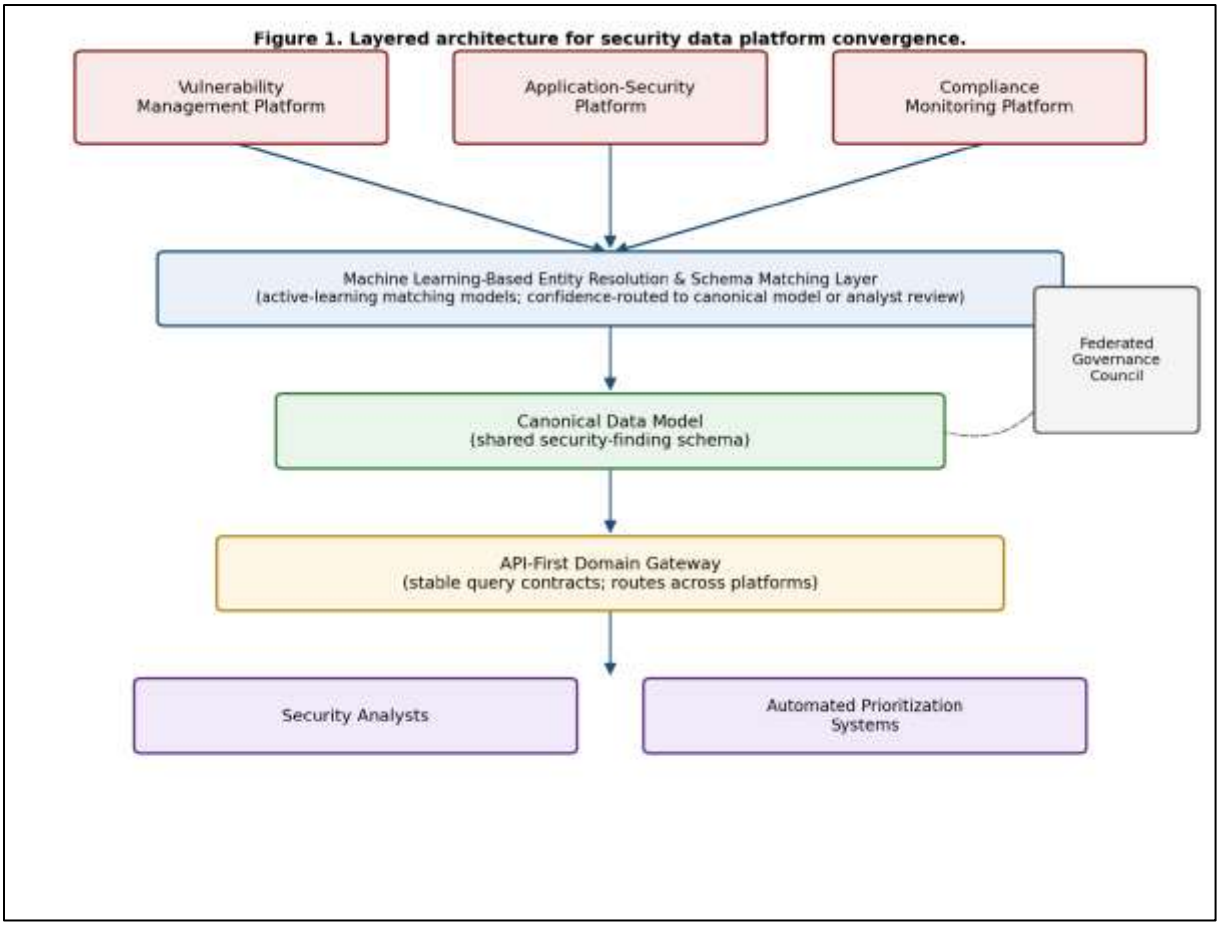

Figure 1. Layered architecture for security data platform convergence.

4.4 API-First Domain Gateways

Once a canonical model and a matching layer exist, API-first domain gateways provide the consumer-facing abstraction that lets convergence proceed without disrupting existing consumers. API-first design treats the interface contract as the primary artifact, agreed before implementation, so that consumers can program against a stable contract regardless of how the underlying platform evolves (Fielding, 2000). In a converging security environment, this means a security analyst or an automated prioritization system queries a single, stable interface for "all critical vulnerabilities on production assets in a given region," and the gateway is responsible for routing that query to whichever platform, or combination of platforms, currently holds the relevant data. Empirical work on microservice API evolution cautions that this stability is not free: once external consumers depend on a published contract, changing that contract carries a coordination cost that teams frequently underestimate, which argues for treating the gateway's contracts as durable design commitments rather than as an implementation detail to be revisited casually (Lercher et al., 2024).

4.5 Progressive Migration Under Federated Governance

The final mechanism addresses how convergence proceeds over time without requiring a disruptive cutover. Fowler's (2004) strangler-fig pattern, originally described for application modernization, offers a template. A progressive migration strategy identifies a high-value integration point and builds a unified capability alongside the existing platforms, then incrementally routes consumers to that capability while monitoring for correctness and performance parity, decommissioning the legacy path only once parity is confirmed so the unified layer gradually envelops the platforms it replaces rather than displacing them in a single event. Applied to security-platform convergence, this pattern maintains continuous security coverage throughout the migration and permits rollback at each stage if the unified capability underperforms. It also distributes organizational change over time rather than concentrating it in a single cutover.

Sustaining this migration over multiple quarters depends on the governance structures described in Section 2.1 and Section 2.2. Data mesh's principle of federated computational governance has cross-cutting decisions about the canonical model and matching thresholds made by a representative body, rather than by a single central team or by each domain team unilaterally. This provides a workable governance pattern, provided the organization also establishes the platform-team capacity that Team Topologies identifies as a common missing ingredient in convergence initiatives that stall (Dehghani, 2022; Skelton & Pais, 2019). Migration research on microservice adoption more broadly finds that organizations underestimate the coordination burden of sustained migration relative to its purely technical difficulty. That caution applies with at least equal force to security platform convergence, where participating teams also carry day-to-day incident-response obligations that cannot be paused for the migration's benefit (Ayas, Leitner, & Hebig, 2023).

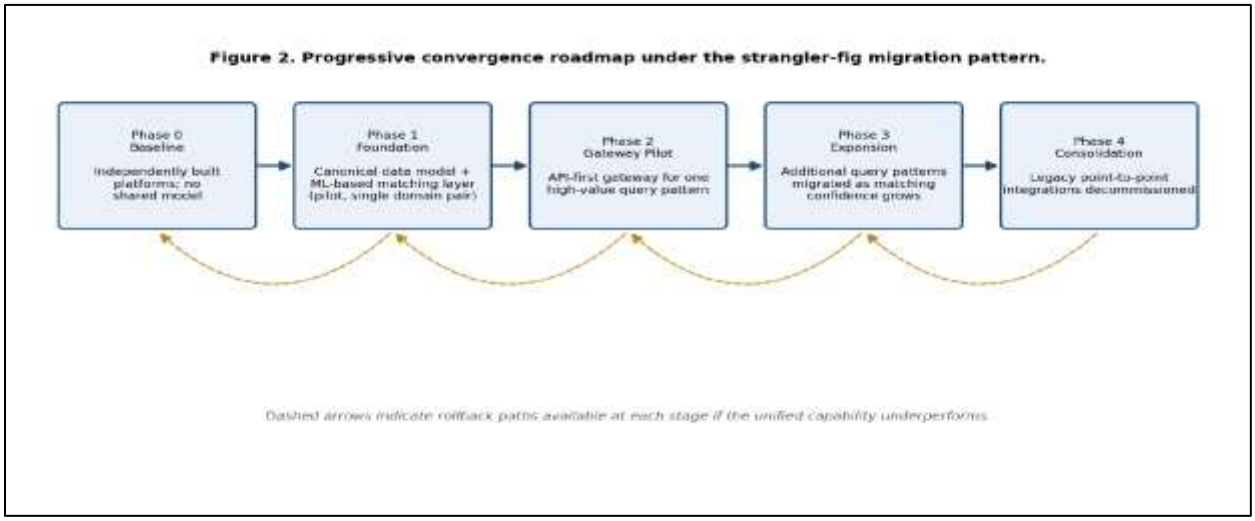

Figure 2. Progressive convergence roadmap under the strangler-fig migration pattern.

5. Illustrative Application

To make the framework concrete, consider a composite scenario representative of the convergence problems documented in Section 2.4. A cloud security organization operates three independently built platforms: a vulnerability management platform that ingests scanner output and tracks remediation status, an application-security platform that records static- and dynamic-analysis findings tied to source code repositories, and a compliance monitoring platform that tracks regulatory obligations against infrastructure configuration. Each platform was built by a different team, at a different time, using different identifiers for the underlying assets each one describes.

Applying the framework begins with the gap-analysis step implicit in Section 4.1: determining, for each pair of platforms, whether the relationship is one of functional overlap that favors eventual consolidation or one of complementary function that favors interoperation through a domain gateway. In this scenario, the vulnerability management and application-security platforms overlap substantially in the assets they describe but differ in the finding types they generate, favoring a canonical-layer approach that correlates their findings without merging the platforms outright. The compliance platform, by contrast, consumes findings from both other platforms as evidence of control status rather than generating comparable findings of its own, favoring a domain-gateway relationship in which it queries the canonical layer rather than being folded into it.

The canonical-layer approach requires resolving, for every asset referenced across the three platforms, whether two differently identified records refer to the same underlying resource the entity resolution problem described in Section 4.3. A machine learning-based matching model trained on the limited set of asset attributes common across the three platforms (network identifiers, ownership metadata, and configuration timestamps) can propose candidate matches with a confidence score, routing high-confidence matches directly into the canonical model and routing low-confidence matches to an analyst queue for manual confirmation rather than resolving them automatically. This confidence-threshold design directly addresses the false-merge risk raised in Section 4.3: a vulnerability finding and an application-security finding are correlated in the canonical model

only once the underlying asset match clears a threshold calibrated to the operational cost of an incorrect merge, not simply the threshold that maximizes overall matching accuracy.

Once the canonical model exists, an API-first domain gateway exposes a unified query surface for example, "all critical findings on internet-facing production assets" that an analyst or an automated prioritization system can use without knowing which of the three platforms originally generated each finding, or how the underlying entities were resolved. The migration to this state proceeds progressively: the gateway is introduced first for a single high-value query pattern, additional query patterns are migrated as confidence in the matching layer grows, and the platforms' original point-to-point integrations are decommissioned only once the gateway demonstrably matches their coverage. Governance of the matching thresholds and the canonical schema's evolution sits with a federated body drawn from each of the three domain teams, consistent with the governance mechanism described in Section 4.5.

The scenario also illustrates why the sequencing of migration matters as much as its endpoint. Introducing the domain gateway before the canonical model and matching layer are reasonably mature would expose consumers to inconsistent results as matching confidence improves underneath them. Introducing the matching layer before any consumer-facing gateway exists, conversely, would leave the organization with a more accurate canonical model but no way for analysts to benefit from it in daily work. The ordering implied by Sections 4.2 through 4.5 canonical model, then matching layer, then gateway, then progressive migration under governance reflects this dependency structure rather than an arbitrary sequence. Organizations that attempt to parallelize these steps to compress a migration timeline should expect to spend at least some of the time saved on reconciling inconsistencies that a more sequential rollout would have avoided.

6. Discussion

6.1 Implications for Security Data Platform Architecture

The framework's central implication is that organizations attempting security-platform convergence without a machine learning-based matching layer are likely to reproduce the same brittleness that motivated convergence in the first place, only one level higher in the architecture. A canonical data model maintained entirely by hand-authored mapping rules still requires an engineer to notice drift and update a rule every time a source platform changes, which is a smaller version of the original integration-tax problem rather than a solution to it. Treating machine learning-based entity resolution and schema matching as a first-class architectural component, rather than as an implementation detail of the canonical model, is what allows convergence to remain sustainable as the number of participating platforms and the pace of their independent evolution both grow.

This has a second, less obvious implication for how convergence initiatives should be staffed. If matching is treated as a one-time data-migration task, it is often assigned to whichever team owns the canonical model at the time of initial integration. If matching is treated as an ongoing architectural capability that must be retrained and monitored as source platforms evolve, it more plausibly belongs with a platform team of the kind Team Topologies describes, staffed with the ability to maintain a machine learning system in production rather than to complete a one-time project (Skelton & Pais, 2019). Organizations that treat convergence as a project with an end date, rather than as a capability with an ongoing operating cost, are likely to underinvest in exactly the mechanism this framework identifies as central.

There is also a cost dimension worth making explicit. Building and operating a machine learning-based matching layer is not free. For an organization with only two platforms and a small number of overlapping entities, hand-authored mapping rules may remain the more economical choice for some time. The argument developed here is not that learned matching is unconditionally preferable, but that its relative advantage grows with the number of platforms, the rate at which their schemas drift, and the engineering time spent maintaining brittle rules. Organizations should therefore expect the point at which a matching layer becomes worth building to depend on their own platform count and rate of change, rather than treating it as a fixed threshold applicable across every organization that reads this framework.

6.2 Limitations of Machine Learning-Based Matching in Security-Critical Contexts

The entity resolution and schema matching literature reviewed in Section 2.3 was developed largely for general-purpose data integration problems, where a false match typically produces a duplicate or merged record that downstream analytics can tolerate or later correct. Security data does not share this tolerance uniformly. A false merge that conflates two distinct vulnerabilities can cause one of them to disappear from an analyst's effective view if the merged record inherits only one finding's remediation status. A false non-match

failing to recognize that two records describe the same asset reproduces the original fragmentation problem at the level of a single entity rather than a whole platform. Neither error is symmetric with the other in operational cost, and neither is well captured by the aggregate accuracy metrics that dominate the entity-resolution benchmarking literature (Binette & Steorts, 2022).

The framework's response to this asymmetry, illustrated in Section 5, is to treat matching confidence as a routing decision rather than a final determination: high-confidence matches populate the canonical model directly, and lower-confidence matches route to human review rather than being resolved automatically in either direction. This design trades some automation for a reduction in the operational cost of matching errors, and the appropriate confidence threshold is itself an organizational judgment about risk tolerance rather than a purely statistical one. It also implies that organizations adopting this framework should expect an ongoing analyst-review workload tied to the matching layer's confidence distribution, not a fully automated pipeline, at least until an organization accumulates enough confirmed cross-platform matches to retrain the model with greater confidence over time.

6.3 Relationship to Prior Convergence Proposals

Prior discussions of security-tool consolidation, most visible in industry analyses of tool sprawl, have generally argued for reducing the number of platforms rather than for architectural mechanisms that let platforms interoperate without reduction (Ponemon Institute, 2020). The framework proposed here treats consolidation as one point on the convergence spectrum described in Section 4.1, not as the necessary goal, on the grounds that some platform boundaries in the illustrative scenario of Section 5 reflect genuine domain specialization rather than accidental duplication. A compliance-monitoring capability, for example, may be better served by a domain-gateway relationship with a canonical vulnerability and application-security model than by full consolidation into it, because its consumers, update cadence, and regulatory obligations differ enough from the other two domains to justify continued architectural separation. This position aligns with data mesh's emphasis on domain-oriented ownership over centralization for its own sake (Dehghani, 2022), while adding the entity resolution and schema matching mechanism that the data mesh literature itself does not specify in detail.

6.4 Implications for Applied Machine Learning Research

Framing security-platform convergence as, in substantial part, an entity resolution and schema matching problem also points to a gap in the applied machine learning literature itself. Most active-learning and deep entity matching research reviewed in Section 2.3 is benchmarked on consumer, bibliographic, or product-catalog datasets, where ground truth is comparatively easy to establish and errors are comparatively cheap (Huang et al., 2023; Jabrane et al., 2024). Security data offers a harder, and for machine learning researchers arguably more interesting, setting. Ground truth for cross-platform matches is scarce precisely because the platforms were never designed to be compared. Label noise is likely to be systematic rather than random, since different platforms encode the same underlying weakness with different severities and taxonomies. And, as Section 6.2 describes, the operational cost of a false positive and a false negative are not merely different in magnitude but different in kind. Extending active-learning and schema-matching methods to be evaluated against these asymmetric, security-specific error costs, rather than against aggregate accuracy alone, is a concrete direction in which the artificial intelligence and machine learning research community could engage directly with the problem this article describes one the security-specific integration literature reviewed in Section 2.4 has not yet pursued in depth.

7. Limitations

This article makes a conceptual and architectural contribution, and its limitations follow directly from that scope. The framework has not been evaluated through a controlled comparison against alternative convergence strategies, nor validated against a benchmark dataset of cross-platform security findings, because no such benchmark specific to security data platform convergence currently exists in the literature reviewed. The illustrative scenario in Section 5 is a composite construction intended to be representative of the convergence problems documented in Section 2.4, not a report of a specific deployment, and readers should treat its outcomes as plausible rather than demonstrated. The confidence thresholds discussed in Section 6.2 are described qualitatively. This article does not propose or validate specific numerical threshold values, since appropriate thresholds depend on an organization's risk tolerance and finding volume in ways that would require empirical calibration against real operational data falling outside the scope of a conceptual framework paper. Finally, the literature underlying the machine learning-based matching component was drawn predominantly from general-purpose entity resolution and schema matching research, rather than from studies conducted specifically on security findings. Section 6.2 treats this as a limitation of the current evidence

base rather than of the framework, and identifies security-specific validation of matching approaches as a priority for future empirical work.

8. Conclusion

Independently built security data platforms fragment for organizational reasons that Conway's Law predicts and that no purely technical intervention resolves on its own. This article has argued that converging such platforms durably, rather than integrating them once and watching the integration decay, depends on treating machine learning-based entity resolution and schema matching as a first-class architectural mechanism alongside a canonical data model, API-first domain gateways, and progressive migration under federated governance. Each mechanism addresses a failure mode the others do not. The canonical model gives platforms a shared representation to converge toward. Machine learning-based matching keeps that representation accurate as source platforms continue to evolve independently. Domain gateways insulate consumers from the details of ongoing migration, and federated governance sustains the initiative past the point where any one team's enthusiasm would otherwise carry it.

The framework's treatment of machine learning-based matching as inherently probabilistic, and therefore requiring a routing decision rather than a fully automated resolution in security-critical contexts, distinguishes its recommendations from general-purpose data-integration practice, where matching errors are more often tolerable. Future work should test this framework's mechanisms against real convergence initiatives, develop benchmark datasets for cross-platform security entity resolution specifically, and establish empirically grounded guidance for setting the confidence thresholds this article has so far only described qualitatively. Doing so would let organizations move from treating platform convergence as a recurring, disruptive project toward treating it as a durable property of how their security data architecture is built.

CRediT Author Contribution Statement

Rahul Kizhakkepachilamakol: Conceptualization, Investigation, Methodology, Writing – Original Draft, Writing – Review & Editing, Visualization. As the sole author, all applicable CRediT roles are attributed to the author; no other contributors are claimed.

Acknowledgments

The author used Claude (Anthropic, claude-sonnet model) to assist with manuscript drafting and language refinement. The author takes full responsibility for the accuracy, originality, and integrity of all content presented in this manuscript, and confirms that all data, analysis, and conclusions reflect independent work and verified sources.

Ethics Statement

Conflict of Interest

The author declares no conflict of interest.

Statement of Human and Animal Rights

This study did not involve human participants, animal subjects, or identifiable personal data requiring ethics board review. The framework and illustrative scenario presented are conceptual and do not draw on any specific organization's proprietary or unpublished data.

Informed Consent

Not applicable.

References

1. Ayas, H. M., Leitner, P., & Hebig, R. (2023). An empirical study of the systemic and technical migration towards microservices. Empirical Software Engineering, 28(4), Article 85. https://doi.org/10.1007/s10664-023-10308-9
2. Balsam, A., Walkowski, M., Nowak, M., Oko, J., & Sujecki, S. (2025). Automatic CVSS-based vulnerability prioritization and response with context information and machine learning. Applied Sciences, 15(16), Article 8787. https://doi.org/10.3390/app15168787

3. Binette, O., & Steorts, R. C. (2022). (Almost) all of entity resolution. Science Advances, 8(12), eabi8021. https://doi.org/10.1126/sciadv.abi8021
4. Conway, M. E. (1968). How do committees invent? Datamation, 14(4), 28–31.
5. Dehghani, Z. (2022). Data mesh: Delivering data-driven value at scale. O'Reilly Media.
6. Evans, E. (2003). Domain-driven design: Tackling complexity in the heart of software. Addison-Wesley Professional.
7. Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation). University of California, Irvine.
8. Fowler, M. (2004). StranglerFigApplication. martinfowler.com. https://martinfowler.com/bliki/StranglerFigApplication.html
9. Goedegebuure, A., Kumara, I., Driessen, S., Van Den Heuvel, W.-J., Monsieur, G., Tamburri, D. A., & Di Nucci, D. (2024). Data mesh: A systematic gray literature review. ACM Computing Surveys, 57(1), 1–36. https://doi.org/10.1145/3687301
10. González-Granadillo, G., González-Zarzosa, S., & Diaz, R. (2021). Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures. Sensors, 21(14), Article 4759. https://doi.org/10.3390/s21144759
11. Huang, J., Hu, W., Bao, Z., Chen, Q., & Qu, Y. (2023). Deep entity matching with adversarial active learning. The VLDB Journal, 32(1), 229–255. https://doi.org/10.1007/s00778-022-00745-1
12. Jabrane, M., Tabbaa, H., Hadri, A., & Hafidi, I. (2024). Enhancing entity resolution with a hybrid active machine learning framework: Strategies for optimal learning in sparse datasets. Information Systems, 125, Article 102410. https://doi.org/10.1016/j.is.2024.102410
13. Lercher, A., Glock, J., Macho, C., & Pinzger, M. (2024). Microservice API evolution in practice: A study on strategies and challenges. Journal of Systems and Software, 213, Article 112110. https://doi.org/10.1016/j.jss.2024.112110
14. Machado, I. A., Costa, C., & Santos, M. Y. (2021). Data mesh: Concepts and principles of a paradigm shift in data architectures. Procedia Computer Science, 196, 263–271. https://doi.org/10.1016/j.procs.2021.12.013
15. Ponemon Institute. (2020). Cyber resilient organization report 2020 (sponsored by IBM Security).
16. Putrama, I. M., & Martinek, P. (2024). Heterogeneous data integration: Challenges and opportunities. Data in Brief, 56, Article 110853. https://doi.org/10.1016/j.dib.2024.110853
17. Ruíz-Ceniceros, J. A., Aguilar-Calderón, J. A., Tripp-Barba, C., & Zaldívar-Colado, A. (2023). Dynamic canonical data model: An architecture proposal for the external and data loose coupling for the integration of software units. Applied Sciences, 13(19), Article 11040. https://doi.org/10.3390/app131911040
18. Santos, P., Abreu, R., Reis, M. J. C. S., Serôdio, C., & Branco, F. (2025). A systematic review of cyber threat intelligence: The effectiveness of technologies, strategies, and collaborations in combating modern threats. Sensors, 25(14), Article 4272. https://doi.org/10.3390/s25144272
19. Skelton, M., & Pais, M. (2019). Team topologies: Organizing business and technology teams for fast flow. IT Revolution Press.
20. Zang, X., Gong, J., Zhang, X., & Li, G. (2023). Attack scenario reconstruction via fusing heterogeneous threat intelligence. Computers & Security, 133, Article 103420. https://doi.org/10.1016/j.cose.2023.103420
21. Zhong, C., Li, S., Huang, H., Liu, X., Chen, Z., Zhang, Y., & Zhang, H. (2024). Domain-driven design for microservices: An evidence-based investigation. IEEE Transactions on Software Engineering, 50(6), 1425–1443. https://doi.org/10.1109/TSE.2024.3385835