

# **AWS Security Hub Master File**

---

## **1. Understanding AWS Security Hub Core Architecture**

---

Short description: Covers the internal service design, evaluators, data ingestion pipeline, workflow engine, control library, and multi-layer architecture.

## **2. Deep Dive into Security Hub Findings: Structure, Normalization, and Data Model**

---

Short description: Covers the AWS Security Finding Format (ASFF), severity scoring, field structure, enrichment, and normalization pipelines.

## **3. How Security Hub Runs Compliance Benchmarks and Control Standards**

---

Short description: Explains benchmarks like CIS, PCI-DSS, AWS Foundational Security Best Practices, control evaluation engine, scheduling, and state storage.

## **4. Internal Evaluation Flow: How Checks Are Triggered, Executed, and Updated**

---

Short description: Focuses on evaluation scheduling, real-time event-driven checks, batch checks, delta checks, and resource re-evaluation mechanics.

## **5. Integration with AWS Services: EventBridge, Config, GuardDuty, Inspector, IAM Access Analyzer**

---

Short description: Covers full ingestion workflow, service-to-Security Hub mapping, event fan-out, and resource/state synchronization.

## **6. How Security Hub Works with Partner Tools and External Security Solutions**

---

Short description: Explains partner integrations, external finding ingestion, bidirectional sync, and third-party product schemas.

## **7. Cross-Account and Cross-Region Aggregation Architecture**

---

Short description: Aggregator account design, region-level flow, administrative accounts, and multi-region rollup mechanics.

## **8. Automation Workflows: EventBridge Rules, Playbooks, and Response Pipelines**

---

Short description: Explains automation architecture, Lambda workflows, Systems Manager Automation, SOAR flows, and orchestrated remediations.

## **9. Custom Insights, Queries, and Advanced Analytical Workflows**

---

Short description: Covers how insights work internally, query model, filters, grouping, and building organization-level analytics.

## **10. Creating and Managing Custom Security Controls**

---

Short description: Covers custom control evaluation logic, JSON structure, event triggers, testing, lifecycle, and deployment across accounts.

## **11. Deep Dive into ASFF Severity, Scoring Models, and Risk Interpretation**

---

Short description: Explains severity models, vendor-provided scoring, confidence levels, evidence blocks, and risk interpretation across multiple services.

## **12. Security Hub and Resource Tagging Strategy for Large-Scale Environments**

---

Short description: Covers tagging models, grouping, environment segmentation, cost attribution, and governance-driven tagging design.

## **13. Operational Behavior: Scaling, Throughput, Throttling, and Event Processing**

---

Short description: Explains ingestion rates, regional scaling boundaries, data plane interactions, and high-volume finding scenarios.

## **14. Security Hub API and CLI Internals for Automation and DevSecOps**

---

Short description: Covers APIs, pagination flows, bulk update patterns, cross-region scripts, and organizational automation design.

## **15. Designing Multi-Account Security Governance with Security Hub**

---

Short description: Explains Org-level deployments, delegated admin patterns, OU-based segmentation, and governance rule distribution.

## **16. Security Hub Cost Structure and Cost Optimization Strategy**

---

Short description: Covers cost drivers, per-finding billing, standards evaluation cost, architectural tuning, and reduction strategies.

## **17. Logging, Monitoring, Auditing, and Forensic Readiness in Security Hub**

---

Short description: Covers CloudTrail interactions, logging flows, evidence collection, historical state analysis, and audit reporting.

## **18. Security Hub in Incident Response and Threat Detection Ecosystem**

---

Short description: Explains integration with SIEM, SOAR, ticketing, IR workflows, and bridging findings into real-time detection pipelines.

## **19. Consolidated Deep Summary of AWS Security Hub Architecture and Operations**

---

Short description: Produces a single long-form, combined summary containing the complete conceptual, architectural, operational, and strategic understanding of the entire topic.

## **20. Common Misconfigurations, Pitfalls, Interview Traps, and Anti-Patterns in Security Hub**

---

Short description: Covers errors, misunderstandings, governance mistakes, wrong aggregation setups, and how to avoid them.

# 1 — Understanding AWS Security Hub Core Architecture

## 1 — Security Hub as a Multi-Layer Security Aggregation System

Security Hub is fundamentally a **multi-layer security aggregation, normalization, and evaluation platform** that centralizes findings from multiple AWS services and third-party tools into one unified model known as the **AWS Security Finding Format (ASFF)**.

– At its core, Security Hub does **not** perform deep packet inspection or host-level telemetry collection; instead, it **aggregates** and **standardizes** the outputs from tools that do.

– The architecture is intentionally designed so that the service sits **above** GuardDuty, Inspector, IAM Access Analyzer, Config, CloudTrail, and numerous external vendors. This high-level position allows Security Hub to maintain a **global, normalized security state** for an entire AWS environment.

Security Hub’s core strength is its ability to unify data that originates from completely different security engines, each producing different types of alerts (resource misconfigurations, malware, anomalous behavior, identity risks, compliance failures, exposure analysis, etc.).

The internal architecture takes these heterogeneous signals and moves them into a **consistent schema, severity system, workflow state, and compliance mapping**.



## 2 — Core Architectural Layers: Deep Breakdown

---

Security Hub's internal architecture can be decomposed into **seven deep layers**, each with distinct responsibilities.

---

### 2.1 — The Ingestion Layer (EventBus + API + Partner Input)

This layer receives all incoming findings and compliance signals.

Three ingestion paths exist:

- **AWS Service Native Pipeline:** GuardDuty, Inspector, Access Analyzer, Config, Macie, Firewall Manager deliver findings directly using a dedicated AWS internal channel.
- **EventBridge-Based Input:** EventBridge routes signals from AWS services or custom applications.
- **Partner / Third-Party Input:** External vendors push ASFF-compliant findings via BatchImportFindings.

This layer is designed for **high throughput**, supporting thousands of findings per second with regional isolation.

Security Hub never stores data cross-region; every region processes its own findings.

---

### 2.2 — The Normalization Engine (ASFF Transformer)

Every incoming finding, regardless of origin, is transformed into **AWS Security Finding Format**.

This includes deep normalization steps:

- Resource mapping
- Severity unification
- Schema expansion
- Field validation
- Evidence consolidation
- Timestamp canonicalization
- Status and workflow alignment

The normalization engine acts like a **translator**, converting differently structured vendor schemas into one uniform model.

---

```
+-----+
|      Normalization Engine      |
+-----+
| - Field Mapping                |
| - Severity Alignment           |
| - Evidence Blocks              |
| - ResourceType Inference       |
| - Timestamp Canonicalizer      |
+-----+
```

---

## 2.3 — Findings Store (Regional, Highly Available Repository)

Security Hub does not use DynamoDB or S3 directly for findings; instead, it uses an internal **managed, replicated, encrypted regional findings store**.

Major characteristics:

- Fully encrypted
- Scalable key-value + document structure
- Optimized for query and update operations
- Stores historical and current findings
- Supports delta updates and patch-based updates

This store enables Security Hub to track:

- Current state
- Workflow changes
- Compliance pass/fail history
- Severity changes
- Remediation progress
- Aggregated view for multi-account setups

---

## 2.4 — Compliance Evaluation Engine (Standards Engine)

This engine runs CIS, PCI-DSS, NIST, FSxBP, and custom standards.

Its deeper structure:

- **Resource Enumerator:** Determines which resources require evaluation
- **Rule Execution Unit:** Runs Config rules or native Security Hub checks
- **Control State Resolver:** Maps rule output to pass/fail status
- **Control Aggregator:** Computes compliance score per standard

- **Evidence Loader:** Tracks evidence for each control

Evaluations run:

- On schedule
- On resource configuration change
- On-demand (manual refresh)
- During cross-account updates

This engine also feeds into the findings store.

---

## 2.5 — Insights Engine (Query/Index Layer)

The Insights Engine builds aggregated, filtered, grouped views on top of findings.

Internally it uses:

- **Indexed search**
- **Faceted grouping**
- **Time-based correlation**
- **Multi-attribute sorting**

This engine allows analysts to query findings by severity, resource type, account, region, control ID, provider, workflow state, and more.

---

## 2.6 — Workflow Engine (State Manager)

This engine manages finding state transitions:

- NEW → NOTIFIED
- NOTIFIED → RESOLVED
- RESOLVED → SUPPRESSED
- SUPPRESSED → ARCHIVED

It records:

- Analyst notes
- Assigned owners
- Remediation timestamps
- Automation-triggered state updates

The workflow engine integrates directly with EventBridge to trigger automations whenever state changes.

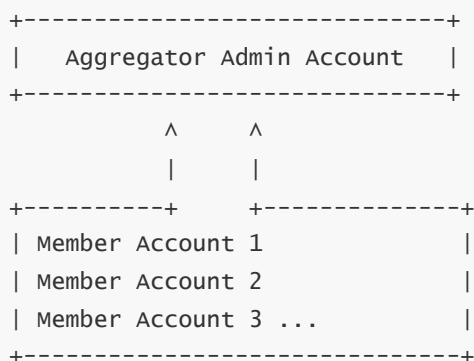
---

## 2.7 — Cross-Account and Cross-Region Aggregation Layer

When configured with delegated admin accounts, the aggregation layer:

- Pulls findings from member accounts
- Normalizes them
- Aggregates them into the admin account
- Syncs workflow states back
- Replicates findings across regions when configured
- Maintains Org config and account metadata

Regional isolation remains—you receive a **multi-region view** only in the aggregator account.



## 3 — Deep Internal Data Flow: End-to-End Trace

To understand the architecture deeply, we trace a full lifecycle:

### Step 1 — Event Occurs

Example: Unauthorized IAM role usage in EC2.

### Step 2 — GuardDuty Generates Finding

Severity, confidence, affected resources determined.

### Step 3 — GuardDuty Pushes Finding to Security Hub

Security Hub's ingestion layer receives it.

### Step 4 — Normalization Engine Transforms Input to ASFF

Fields standardized.



## Step 5 — Findings Stored in Regional Repository

Stored as structured ASFF document.

## Step 6 — Compliance Engine Re-Evaluates Controls

If required, updated compliance results are generated.

## Step 7 — Insights Engine Updates Aggregated Views

Your dashboards immediately reflect changes.

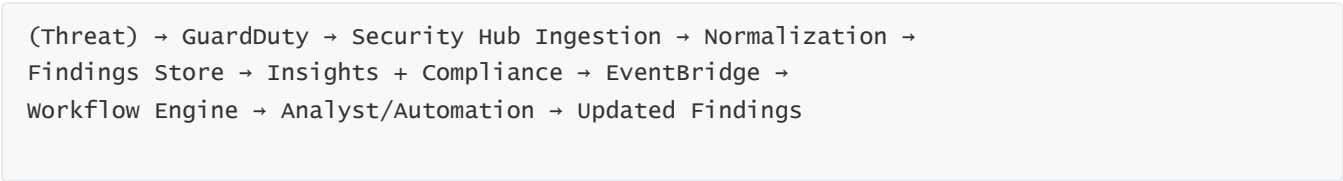
## Step 8 — EventBridge Emits a Finding Change Event

Automation and SOAR pipelines trigger.

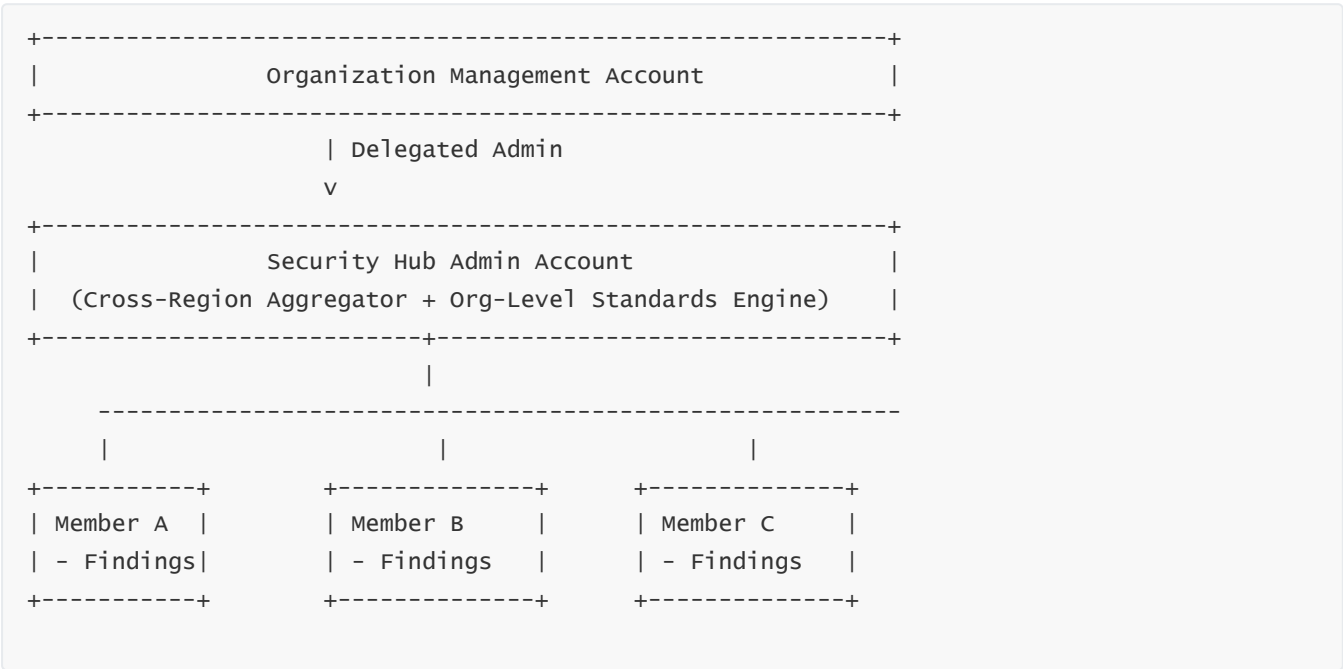
## Step 9 — Workflow Engine Tracks State Changes

Analyst or automation-driven remediation updates state.

This continuous loop maintains an always-up-to-date view of security posture.



# 4 — Security Hub Architecture in a Full Multi-Account Scenario



This architecture supports:

- Hierarchical governance
  - Centralized compliance
  - Enterprise-level dashboards
  - Automated cross-account remediation
  - Unified workflow management
- 

## 5 — Architectural Strengths and Guarantees

---

Security Hub’s design provides:

- **High availability** through regional isolation and internal replication
  - **Strong consistency** for finding state
  - **Deterministic compliance calculation**
  - **Scalable ingestion** supporting high-volume environments
  - **Unified risk model** across dozens of service types
  - **Extensibility** through ASFF for partner integrations
  - **Operational simplicity** via EventBridge and workflow automation
- 

## 2 — Deep Dive into Security Hub Findings: Structure, Normalization, and Data Model

---

### 1 — What exactly is an AWS Security Hub “finding”?

A Security Hub “finding” is a **normalized security record** that describes a specific security-relevant condition in your environment, using a consistent schema called the **AWS Security Finding Format (ASFF)**.

- Conceptually, you can think of a finding as a **single security event or condition**, captured with full context: which resources are involved, what is wrong, how severe it is, when it happened, who reported it, and what you can do about it.
- Unlike raw alerts from individual services (GuardDuty, Inspector, IAM Access Analyzer, external scanners), a Security Hub finding is already **normalized**, meaning it has been transformed into a standard shape so that dashboards, queries, and automation can treat all findings in the same way.

This unified record is the core data unit of Security Hub. Every compliance control failure, every GuardDuty detection, every Inspector vulnerability, and every partner alert becomes an ASFF finding with a consistent structure.

---

### 2 — High-level structure of an ASFF finding

At a high level, an ASFF finding is a **JSON document** with several top-level sections. These sections follow a logical structure:

- “Who and what produced this finding?” (product and provider metadata)
- “What is the security problem?” (title, description, type, severity, compliance, threat Intel)
- “Where in AWS did this occur?” (account, region, partition, resource details)
- “When did the important events happen?” (discovered, first seen, last updated)
- “What is its current status in the workflow?” (workflow state, record state)
- “What should we do about it?” (remediation, recommended actions, related tickets)

A simplified conceptual view looks like this:

```
Finding
├─ Product / Provider Metadata
│  ├─ ProductArn
│  ├─ AwsAccountId
│  ├─ GeneratorId
│  └─ Types[]
│
├─ Identification & Description
│  ├─ Id
│  ├─ Title
│  ├─ Description
│  └─ FindingProviderFields
│
├─ Severity & Classification
│  ├─ Severity
│  ├─ Confidence
│  └─ Criticality
│
├─ Affected Resources
│  └─ Resources[]
│
├─ Temporal Data
│  ├─ FirstObservedAt
│  ├─ LastObservedAt
│  ├─ CreatedAt
│  └─ UpdatedAt
│
├─ Compliance & Security Context
│  ├─ Compliance
│  ├─ Vulnerabilities[]
│  ├─ Malware[]
│  └─ ThreatIntelIndicators[]
│
├─ Workflow & Record State
│  ├─ workflow
│  └─ RecordState
│
└─ Remediation & Extras
   ├─ Remediation
   ├─ RelatedFindings[]
   └─ UserDefinedFields / Note etc.
```

Each of these “sections” is a cluster of fields designed for a specific use: dashboards, automation, compliance reporting, investigations, and so on.

---

### 3 — Core identification and producer fields

#### 3.1 — ProductArn

The `ProductArn` identifies **who produced the finding**.

- For GuardDuty, it is an ARN that encodes the GuardDuty detector in that region.
- For Inspector, it identifies the Inspector product in that region.
- For partner tools, it identifies the partner’s Security Hub-integrated product.

This makes it possible to filter or group findings by source (e.g., show only GuardDuty, show only Inspector, show only a specific partner).

---

#### 3.2 — AwsAccountId

The `AwsAccountId` indicates **which AWS account the finding is about** (the “victim” or environment owner, not necessarily where the product is deployed).

- In a multi-account Security Hub setup, the **admin/aggregator account** will be receiving findings where `AwsAccountId` is the member account ID, not the admin’s own ID.
  - This is crucial for cross-account aggregation, compliance, and billing analyses.
- 

#### 3.3 — Id and GeneratorId

- `Id` is the **unique identifier of the finding instance** within Security Hub for that account and region. It is used to reference and update the finding over time.
- `GeneratorId` is the **logical source inside the product** that generated the finding:
  - For GuardDuty, GeneratorId might represent the specific detection type (e.g., `Recon:EC2/PortProbeUnprotectedPort`).
  - For Config-based controls, GeneratorId might reference a Config rule or a Security Hub control ID.

`GeneratorId` is especially important for deduplication and correlation, because multiple findings that come from the same rule/detector but different resources can be grouped or reasoned about together.

---

### 4 — Title, Description, Types: How Security Hub classifies the problem

#### 4.1 — Title and Description

- `Title` provides a **short, human-friendly label** describing the issue (e.g., “S3 Bucket Allows Public Read Access”).
- `Description` gives a **longer explanation** that may include what was found, why it is important, and sometimes high-level remediation hints.

These fields are crucial for dashboards and human consumption; they are what analysts read first in the console.

---

## 4.2 — Types (threat taxonomy classification)

`Types` is an **array of strings** representing **categorical classifications** for the finding, following a taxonomy like:

- `Software and Configuration Checks/Industry and Regulatory Standards/CIS AWS Foundations Benchmark`
- `TTPs/Initial Access`
- `TTPs/Exfiltration`

This classification allows Security Hub and external tools to group findings based on **threat category, technique, or compliance domain**, enabling powerful filters and analytics (for example, “show me all exfiltration-related issues across all accounts”).

---

## 5 — Deep dive into Severity, Confidence, and Criticality

### 5.1 — Severity object

The `Severity` structure contains fields such as:

- `Label` (e.g., `INFORMATIONAL`, `LOW`, `MEDIUM`, `HIGH`, `CRITICAL`)
- Possible numeric score (like `Normalized`, or provider-specific numeric severity)

The key idea is that Security Hub **standardizes severity representations** from multiple producers into consistent labels so that you can sort and prioritize across GuardDuty, Inspector, partner tools, etc., without understanding each tool's proprietary scoring system separately.

---

### 5.2 — Confidence and Criticality

- **Confidence** represents how strongly the product believes this finding is accurate and not a false positive. Higher confidence means the provider has strong evidence and the event is likely real.
- **Criticality** reflects **how important the underlying resource is** to your business or to the environment. For example, a HIGH criticality might be used for production workloads or sensitive data systems.

By combining `Severity`, `Confidence`, and `Criticality`, Security Hub enables more nuanced ranking:

– A MEDIUM severity on a HIGH criticality asset with high Confidence might, in practice, be treated similarly or higher than a HIGH severity finding on a low criticality asset.

---

## 6 — Resource representation: The heart of ASFF

### 6.1 — Resources[] array

Every finding carries a `Resources` array that represents **one or more entities affected by the finding**.

Each entry in this array might represent:

- An EC2 instance

- An S3 bucket
- An IAM role or user
- A VPC, security group, subnet
- A load balancer, RDS instance, Lambda function, etc.

Each resource object typically includes:

- `Type` — the standardized resource type (e.g., `AwsEc2Instance`, `AwsS3Bucket`, `AwsIamRole`)
- `Id` — usually an ARN or ARN-like identifier
- `Region` — the resource's region
- `Partition` — `aws`, `aws-cn`, `aws-us-gov`
- `Details` — nested data specific to that resource type (tags, configuration properties, encryption status, security groups, etc.)

This `Resources[]` structure is what allows Security Hub to group and correlate findings per resource and to show “all problems affecting this resource” in one place.

---

## 6.2 — Resource details and configuration snapshot

For resource types with rich configuration (e.g., `AwsEc2Instance`, `AwsS3Bucket`), the `Details` section can include:

- Network interfaces, public IPs, and security groups for EC2
- Bucket policies, ACLs, encryption configuration, and access logging flags for S3
- Trust policy and inline policies for IAM roles

This effectively gives you a **partial configuration snapshot** at the time of the finding, which is very useful for investigations and historical analysis.

```
Resource
├─ Type: AwsS3Bucket
├─ Id: arn:aws:s3:::my-logs-bucket
├─ Region: us-east-1
├─ Partition: aws
└─ Details
    ├─ BucketVersioningConfiguration
    ├─ ServerSideEncryptionConfiguration
    ├─ LoggingConfiguration
    └─ PublicAccessBlockConfiguration
```

This structure allows automation and humans to see not only *that* a bucket is public, but **exactly how it is configured** and what other misconfigurations may co-exist.

---

## 7 — Temporal model: when did this happen?

Security Hub uses multiple timestamps to capture the lifecycle of the condition described by the finding:

- `FirstObservedAt` — the time the provider **first saw** this issue.
- `LastObservedAt` — the most recent time the issue was observed or reconfirmed.
- `CreatedAt` — when the finding record was **created in Security Hub**.
- `UpdatedAt` — when the finding was **last updated** (e.g., severity change, workflow state change, note added).

This temporal model allows us to distinguish **stale** issues from **recent** ones and understand if the condition is ongoing or was a short-lived anomaly.

---

## 8 — Compliance, vulnerabilities, and threat intelligence sections

### 8.1 — Compliance block

For findings that come from Security Hub standards (e.g., AWS Foundational Best Practices, CIS Benchmarks), or from tools that map to compliance frameworks, the `Compliance` section may include:

- `Status` — `PASSED`, `FAILED`, `WARNING`, `NOT_AVAILABLE`
- `RelatedRequirements` — which CIS/NIST/PCI etc. controls this relates to
- Possibly compliance check identifiers

This is the **bridge between Security Hub findings and compliance reporting**. A single detailed finding may support multiple compliance requirements.

---

### 8.2 — Vulnerabilities[]

For Inspector or partner vulnerability scanners, a finding may contain a `Vulnerabilities` array, with each entry holding:

- CVE identifiers
- CVSS scores
- Affected package names and versions
- Fix versions
- Related URLs and references

This effectively embeds the **vulnerability intelligence** directly within the Security Hub finding. Automation tools can use this to prioritize patching and remediation based on CVSS or other vulnerability attributes.

---

### 8.3 — Malware[] and ThreatIntelIndicators[]

- `Malware[]` describes detected malware families, names, and behaviors.
- `ThreatIntelIndicators[]` provides threat intel matches, such as malicious IPs, domains, or file hashes.

These sections are particularly relevant for GuardDuty and external threat detection tools that feed into Security Hub.

---

## 9 — Workflow and record state: how Security Hub tracks progress

## 9.1 — Workflow

The `workflow` object describes the **current status of investigation/remediation**. Typical labels include:

- `NEW` — just created, not yet processed by an analyst or automation.
- `NOTIFIED` — someone or some system has been notified.
- `RESOLVED` — the issue has been fixed.
- `SUPPRESSED` — intentionally ignored or accepted risk.

Security Hub's workflow status is separate from the existence of the underlying condition. For example, a misconfiguration may still exist technically, but you might have suppressed it for a specific reason (e.g., intentional exposure, test environment).

---

## 9.2 — RecordState

`RecordState` typically captures whether the finding is considered “active” by Security Hub:

- `ACTIVE` — relevant and visible.
- `ARCHIVED` — logically archived; no longer returned in default queries.

`RecordState` is mainly about **visibility and retention** in Security Hub, while `workflow` is about **investigation and remediation state**. Together, they form a two-dimensional state model: one dimension for “is this still logically active?” and one dimension for “how far along is the remediation or review?”.

---

## 10 — Remediation, related findings, and user-defined fields

### 10.1 — Remediation block

The `Remediation` section often contains:

- `Recommendation` text (what you should do).
- Possibly URLs or documentation pointers.

This allows tools and analysts to build **automated or semi-automated playbooks** that read recommended actions and perform them via Lambda, SSM, or external SOAR platforms.

---

### 10.2 — RelatedFindings and UserDefinedFields

- `RelatedFindings[]` allows linking multiple findings together (for example, a GuardDuty detection and a Config control failure about the same resource).
- `UserDefinedFields` gives you a place to store your own key-value metadata, such as ticket IDs, owner teams, environment tags, business impact tags, etc.

This makes ASFF **extensible**, allowing you to enrich findings with internal context and tracking identifiers.

---

## 11 — Normalization pipeline: how raw events become ASFF findings

From an architectural perspective, there is a **normalization pipeline** between the producing service and Security Hub's stored finding:



```
Raw Detection / Event
|
v
Provider Internal Model (GuardDuty / Inspector / Partner)
|
v
Security Hub Ingestion Adapter
|
v
ASFF Normalization Engine
|
v
Security Hub Finding (ASFF) in Findings Store
```

Key steps in the normalization pipeline:

- Field mapping: map provider fields into standardized ASFF fields.
- Resource modeling: construct `Resources[]` entries with proper types and details.
- Severity alignment: provider-specific severity converted into standard labels.
- Compliance mapping: link to compliance requirements when relevant.
- Metadata enrichment: add timestamps, product ARNs, generator IDs, and record IDs.

This pipeline allows Security Hub to **merge and correlate** findings from very different systems into one coherent data model.

---

## 12 — Partial updates, deduplication, and lifecycle management

Providers and partners do not always send a full new finding every time something changes. Instead, they often perform **partial updates**:

- Security Hub uses the `Id` and `ProductArn` as key identifiers for a finding.
- When new data arrives, Security Hub can **patch** existing findings, updating only specific fields (e.g., severity, workflow, last observed time).

Deduplication logic ensures that repeated detections of the same ongoing condition **update the existing finding** instead of creating thousands of duplicates.

Lifecycle-wise, a finding may:

- Start as NEW / ACTIVE.
  - Be updated many times as new evidence arrives.
  - Move through workflow states as it is investigated.
  - Be resolved and eventually archived when the condition is permanently fixed or accepted.
- 

## 13 — Organization-level and cross-account implications of the data model

In a **multi-account, multi-region** deployment, the ASFF structure remains the same, but there are strong organizational implications:

- `AwsAccountId` is used heavily in queries, insights, and dashboards to separate or group findings by environment (prod vs dev, OU-based segmentation, etc.).
- Aggregator accounts store and display findings from many accounts, but the **identity and resource references still belong to the original member account**.
- Resource ARNs combined with `AwsAccountId` allow you to pivot from a central Security Hub console into the actual resource in its home account.

The data model is therefore carefully designed to preserve correct **ownership, scoping, and access context** while allowing central governance.

---

## 14 — How the ASFF model supports analytics, insights, and automation

Because findings follow a consistent JSON schema with well-defined fields, the ASFF model is ideal for:

- **Insights:** Security Hub can build powerful queries grouping by severity, account, resource type, standard, or workflow state, using standardized fields.
- **EventBridge automations:** Rules can trigger specifically on `Severity.Label`, `Compliance.Status`, `Workflow.Status`, `Resource.Type`, or `AwsAccountId`, enabling precise remediation playbooks.
- **Exports to SIEM/SOAR:** External tools can parse ASFF reliably, since they know where to find core metadata, resources, and security context.
- **Custom controls and advanced compliance:** Custom rules can output findings that seamlessly fit into the same model, so your own checks behave just like AWS-managed ones.

In other words, the ASFF data model is designed not just for display, but for **machine-driven processing, correlation, and orchestration**.

---

## 15 — Mental model summary of a Security Hub finding

You can build a simple mental model of an ASFF finding as:

```
"A normalized, versioned, security record that says:  
- This provider (ProductArn) in this account (AwsAccountId)  
  detected this problem (Title/Description/Types)  
  affecting these resources (Resources[])  
  with this severity and confidence (Severity/Confidence)  
  for these compliance requirements (Compliance)  
  at these times (FirstObservedAt...UpdatedAt)  
  and it is currently here in our process (Workflow/RecordState)  
  with this recommended fix (Remediation)  
  and this additional context (Vulnerabilities, Malware,  
  ThreatIntelIndicators, UserDefinedFields)."
```

This is the “unit of truth” that Security Hub uses everywhere — dashboards, insights, automation, cross-account governance, and compliance reporting.

---

# 3 — How Security Hub Runs Compliance Benchmarks and Control Standards

---

## 1 — What is a “standard” and a “control” inside Security Hub?

When we talk about Security Hub and “compliance,” we are really talking about a **hierarchy of standards and controls** that the service uses to evaluate your AWS environment.

- A **standard** is a **collection of security controls** that collectively implement a particular benchmark or framework. Examples: **AWS Foundational Security Best Practices (FSBP)**, **CIS AWS Foundations Benchmark**, **PCI DSS**, and other industry standards.
- A **control** is a **single, testable security requirement** inside a standard. For example, “S3 buckets should not allow public read access,” or “CloudTrail must be enabled in all regions.”

So internally, Security Hub treats your compliance configuration as a large **graph of standards → controls → checks → resources**. Each control has its own logic (how to evaluate), mapping (which resources it applies to), and state (PASSED/FAILED/etc.), and the standard simply groups them together and defines how to roll up control states into an overall compliance posture.

---

## 2 — Internal data model for standards and controls

Security Hub represents each standard and control with **metadata objects** and **evaluation logic descriptors**, not just human-readable descriptions.

- For each **standard**, Security Hub stores: unique ID, version, list of controls, enablement status per account and region, and configuration (for example, which controls are disabled).
- For each **control**, Security Hub stores: unique control ID, the underlying implementation type (Config rule, native Security Hub rule, or integration-based rule), the set of **resource types** it evaluates, and the mapping to external frameworks (e.g., NIST, PCI, ISO).

This metadata plus evaluation logic is what drives the **Compliance Engine**. When you enable a standard, Security Hub doesn't just show a static checklist; it actually **activates the underlying evaluation logic** that runs continuously against your environment.

---

## 3 — Standards enablement: how Security Hub “turns on” benchmarks

When you enable a standard (for example, AWS Foundational Security Best Practices) in a given account and region, Security Hub performs several steps internally:

### 3.1 — Register the standard and control set for that account-region

- Security Hub marks the standard as “enabled” for that specific account and region.
- It loads the **list of controls** belonging to that standard into that account-region's configuration.

### 3.2 — Activate underlying rules (often via AWS Config or native engines)

- Many controls are implemented using **AWS Config rules**, so Security Hub will either **create new managed Config rules** or link to existing ones.
- Other controls are implemented by **native Security Hub evaluators** that query resource configuration directly using internal APIs or data from Config, CloudTrail, or other sources.

### 3.3 — Initialize compliance state

- For each control, Security Hub sets an initial compliance status ( `NOT_AVAILABLE` or similar) until the first evaluation run completes.
- A scheduled or immediate **initial evaluation** is triggered so that the compliance dashboards start populating quickly after standard activation.

---

## 4 — Control evaluation types: Config-based, native, and integration-based

Internally, Security Hub supports multiple “flavors” of controls, depending on which underlying service or logic they rely on:

### 4.1 — Config-based controls

These controls rely on **AWS Config** evaluations.

- Security Hub ties a control to a specific **Config rule** (managed or custom).
- Whenever Config evaluates resources for that rule, it sends **Config evaluation results** (COMPLIANT / NON\_COMPLIANT) which Security Hub converts into **control-level compliance states** and, where appropriate, into **findings**.
- This means Security Hub doesn’t need to reinvent resource-change detection or drift tracking; it leverages Config’s configuration timeline and evaluation engine.

### 4.2 — Native Security Hub controls

Some controls are evaluated directly by Security Hub using internal logic.

- These controls query AWS APIs, or internal data sources, to check configuration states (e.g., “Is Security Hub enabled across all regions?” or certain global-best-practice checks).
- They run in **scheduled waves** or are triggered by **events** (such as resource changes or new findings) depending on the control’s design.

### 4.3 — Integration-based controls

Certain compliance requirements depend on other security services’ outputs.

- For example, a standard might require that **GuardDuty be enabled**, or that **Inspector coverage** is sufficient.
- In such cases, Security Hub evaluates a control based on the configuration or findings of those services, rather than direct resource checks.

Conceptually, a control is simply a **wrapper** around whichever evaluation mechanism applies:

```
Standard
├─ Control C1
│   ├─ Type: Config-based / Native / Integration-based
│   ├─ Target Resource Types
│   └─ Evaluation Logic (rule, query, or condition)
```

---

## 5 — The compliance evaluation engine: control lifecycle and state machine

The **Compliance Engine** in Security Hub handles control evaluations and state transitions. For each control, the engine maintains a **state machine** that tracks whether that control is passing, failing, or not applicable.

At a high level, per control per account-region, you have states like:

- **PASSED** — all applicable resources meet the requirement.
- **FAILED** — one or more resources are out of compliance.
- **WARNING** — ambiguous result or partial coverage.
- **NOT\_AVAILABLE** — not enough information to evaluate.

Internally, it works like this:

```
Resource Config / Events
  |
  v
Underlying Check (Config rule / native evaluator / integration)
  |
  v
Raw Evaluation Result (COMPLIANT / NON_COMPLIANT / N/A / ERROR)
  |
  v
Control State Resolver
  |
  v
Control Compliance State (PASSED / FAILED / WARNING / NOT_AVAILABLE)
  |
  v
Optional Control Finding (+ Compliance block)
```

The **Control State Resolver** is responsible for mapping potentially complex raw results (e.g., partial resource coverage, transient errors, per-resource statuses) into a single control-level status.

---

## 6 — Trigger mechanisms: when does Security Hub re-run controls?

Security Hub doesn't simply run compliance checks once and then forget. It uses multiple trigger mechanisms:

### 6.1 — Scheduled evaluations

- Many controls are evaluated on a **periodic schedule** (e.g., every few hours).
- During these scheduled runs, the engine enumerates applicable resources and re-evaluates them, updating control states and generating or updating findings.

### 6.2 — Event-driven evaluations

- For Config-based controls, every time **AWS Config sees a configuration change**, it may immediately evaluate the relevant rule and send results.
- For some native controls, Security Hub listens to **events from other services** or internal signals (e.g., guardrails enabling/disabling) to trigger re-checks.

### 6.3 — On-demand refresh

- In the console, you can manually **re-run a control** or standard. This triggers an immediate evaluation cycle for that control's logic.

Together, these mechanisms ensure that compliance posture stays **reasonably close to real time**, especially for high-sensitivity settings.

---

## 7 — Resource scoping: which resources does a control apply to?

Each control has an internal notion of **scope**: which resource types, accounts, and regions are in the evaluation set.

- **Resource type scope**: Some controls apply only to S3 buckets, some to IAM users or roles, some to RDS DB instances, etc.
- **Account scope**: In a single account, all resources of the eligible type can be evaluated. In an organization-wide setup, each **member account** evaluates its own resources, and results are aggregated.
- **Region scope**: Many controls are region-specific; for example, “CloudTrail must be enabled in all regions” is evaluated **per region** and then aggregated to determine if you truly have global logging.

Internally, the Compliance Engine uses **resource enumerators** that know how to list all relevant resources for a given control and then feed them into the underlying check logic.

---

## 8 — From control evaluation to findings: the compliance → finding bridge

When a control runs and finds non-compliant resources, Security Hub often generates or updates **findings** that represent those failures. The flow looks like this:

```
Control Evaluation
|
v
Per-resource status (compliant / non-compliant)
|
v
For each non-compliant resource:
  Generate/Update ASFF Finding
    ├── ProductArn = securityhub
    ├── GeneratorId = <StandardID/ControlID>
    ├── Compliance.Status = FAILED
    ├── Resources[] = <non-compliant resource(s)>
    └── Severity / Title / Description
```

- Each such finding includes a **Compliance block** that states whether the control is PASSED/FAILED for that resource and which **requirements** (CIS, PCI, etc.) it corresponds to.
- Security Hub then uses these findings to power **dashboards, insights, automation, and reports**.

So, the standard/control layer is not only about checklists; it's a **finding generator** that continuously feeds normalized ASFF findings into the system.

---

## 9 — Rolling up control results into standard-level scores

At the **standard level**, Security Hub computes compliance summaries based on the status of all controls under that standard.

Conceptually:

```
Standard S
├─ Controls: C1, C2, ..., Cn
│   └─ Each has a current status (PASSED/FAILED/...)
└─ Standard Summary = Aggregation(C1..Cn)
```

- Security Hub computes metrics like **percentage of controls passed**, **number of FAILED controls**, etc.
- These are shown in the console as **compliance percentages** and summary charts.
- In more advanced views, you can drill down into **per-control and per-account** compliance states.

For large organizations, this rollup is essential: instead of manually tracking hundreds of individual checks, you look at a small set of **standard-level posture indicators**, then drill down to the problematic controls and resources.

---

## 10 — Multi-account and organization-level compliance evaluation

In an **AWS Organizations** setup with **Security Hub delegated admin**, compliance is evaluated both **locally** and **centrally**:

- Each **member account** runs its own control evaluations (Config rules, native checks, etc.) within its own region.
- Those evaluation results and generated findings are then **aggregated** into the **admin account**, which can show consolidated compliance for all accounts.

Architecturally, it looks like this:

```
[Member Account 1]  [Member Account 2]  [Member Account 3]
|                   |                   |
| Control Results   | Control Results   |
+-----+-----+-----+-----+
|
| v
Security Hub Admin Account
├─ Aggregated Control States
└─ Org-Level Standard Summaries
```

The admin account does not re-evaluate the controls; it **trusts** the member accounts' evaluations and simply aggregates their results. This respects the principle that each account owns its own resource configuration and evaluation, while the admin account owns **governance visibility**.

---

## 11 — Disabling controls and partial standard enablement

In real-world environments, you often do not want **every** control from a standard. Security Hub allows you to disable individual controls:

- When a control is **disabled**, Security Hub stops evaluating that control for that account/region and no longer uses it in compliance rollups.
- Existing findings may be **archived or left as historical** depending on your workflow, but new evaluations will not run for that control.

This is especially important when:

- A control does not apply to a specific environment (e.g., you do not use certain services).
- You have an **exception policy** or alternative compensating control outside AWS.

So standards in Security Hub are **not all-or-nothing**; they can be fine-tuned control by control.

---

## 12 — Versioning and updates of standards

Benchmarks like CIS and AWS FSBP evolve over time. When AWS updates a standard:

- Security Hub introduces a **new version** of the standard (for example, v1.0 to v1.1).
- Each version may have **added, removed, or modified controls**.
- You can often see and choose which version you are enabling.

Internally, versioning means:

- A different control-set definition.
- Possibly different evaluation logic for some controls.
- Different mappings to external requirement IDs.

This versioning ensures that **your compliance posture can track the evolution of best practices**, and that reports can correctly claim compliance against specific versions of standards.

---

## 13 — Error handling, NOT\_AVAILABLE state, and partial evaluation

Not every control evaluation is always successful. Sometimes:

- Resources are temporarily unreachable.
- Underlying APIs fail or throttle.
- A service is not enabled in that region.

In such cases, the Compliance Engine may set a control or resource to `NOT_AVAILABLE` or some warning-like state rather than **falsely** marking it as PASSED or FAILED. This is a crucial nuance for serious compliance work:

- A `NOT_AVAILABLE` status indicates **lack of evidence**, not safety.
- You might treat such controls similarly to failures in your operational processes, because you cannot prove compliance.

Internally, the engine distinguishes between:

- **Evaluation failure** (system problem).
- **Non-applicability** (control does not logically apply to environment).



These differences affect how the control contributes to the standard's summary metrics.

---

## 14 — How compliance benchmarks integrate with Insights and dashboards

Because compliance evaluations generate **standardized findings** and control statuses, Security Hub can feed that data into:

- **Compliance views:** dedicated dashboards showing per-standard and per-control compliance.
- **Insights:** queries like “show all FAILED controls from CIS in production accounts” or “show all non-compliant S3-related controls.”
- **Trends over time:** by tracking updated timestamps and control state changes, you can see whether posture is improving or degrading.

Under the hood, this works by storing control-compliance information in the same **findings store** and using the **Insights Engine** to index and query it, just like any other Security Hub finding.

---

## 15 — Triggering automations based on compliance failures

Compliance controls are often used as triggers for **automatic remediation and enforcement**. A common pattern:

```
Control Fails → Security Hub generates/updates FAILED finding
                → EventBridge rule matches on Compliance.Status = FAILED
                → Invoke Lambda / SSM Automation / SOAR playbook
                → Remediate (e.g., block public access, enable logging)
                → Update finding workflow.Status and possibly resolve
```

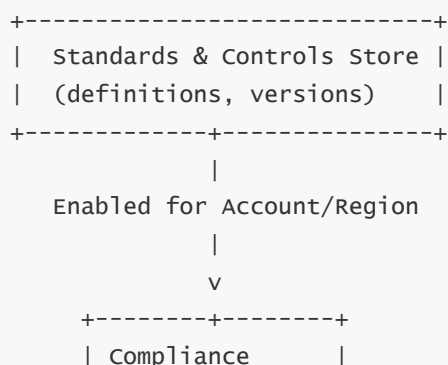
Because control failures produce ASFF findings with consistent **Compliance** and **Resource** sections, you can:

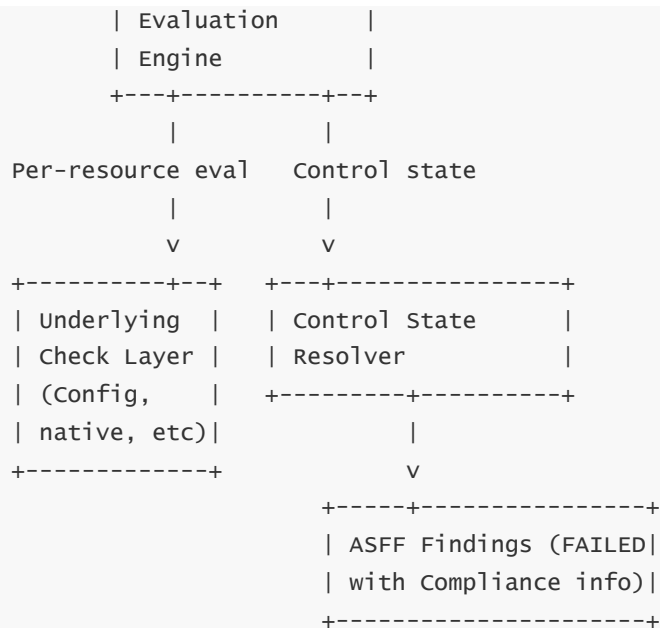
- Write EventBridge rules matching on `Compliance.Status = FAILED`, `Severity.Label`, `Resource.Type`, or specific `StandardsArn` / `ControlId`.
- Route them into automated **fix-it functions**.

This turns compliance from a purely reporting-oriented feature into a **real-time enforcement and remediation mechanism**.

---

## 16 — Architectural diagram: compliance evaluation and finding generation





- The **Standards & Controls Store** holds the definitions.
- The **Compliance Engine** drives evaluation, using the appropriate underlying check layer.
- The **Control State Resolver** maps raw results to control status.
- Finally, **ASFF findings** are created or updated and pushed into the findings store for use in dashboards, insights, and automation.

## 17 — Practical mental model for how Security Hub runs benchmarks

You can compress all of this into one mental sentence:

“When I enable a standard in Security Hub, I am telling the service to continuously run a curated set of controls (implemented via Config, native logic, and integrations) across my accounts and regions, maintain a live state machine for each control, translate failures into ASFF findings with compliance metadata, and then roll those up into standard-level scores that I can query, visualize, and automate against.”

This is how Security Hub turns abstract documents like “CIS benchmark” or “AWS best practices” into a **live, continuously evaluated, machine-actionable compliance system** inside AWS.

# 4 — Internal Evaluation Flow: How Security Hub Checks Are Triggered, Executed, and Updated

# 1 — Understanding the Core Evaluation Lifecycle: From Trigger → Evaluation → Update → Finding

---

To understand how Security Hub evaluates resources, we need to understand that Security Hub does *not* operate like a traditional scheduled scanner that runs “every X hours.” Instead, Security Hub uses a **hybrid internal evaluation model**, composed of:

- **Event-based triggers** (real-time resource changes, new findings, config drift)
- **Incremental scheduled waves** (periodic reevaluations for consistency)
- **Dependency-driven triggers** (events from AWS Config, GuardDuty, Inspector, IAM Access Analyzer, etc.)
- **On-demand evaluations** (manual refresh, control override, remediation validation)
- **Continuous delta-based updates** (partial evaluation for changed resources only)

The combination of these mechanisms ensures that Security Hub always maintains a **near real-time, authoritative compliance and security posture** for your accounts and regions.

This entire lifecycle revolves around the “Evaluation Pipeline,” which maintains the state of:

- Controls
- Findings
- Resources
- Standards
- Cross-account aggregated state

Security Hub continuously moves data through this pipeline to ensure that your compliance posture reflects the actual configuration and threat landscape.

---

## 2 — The Internal Evaluation Architecture: Multi-Stage Processing Pipeline

---

```
+-----+
|           Security Hub Evaluation Pipeline           |
+-----+
| 1. Trigger Layer                                     |
| 2. Resource Enumeration                             |
| 3. Underlying Checks (Config / Native / Integration) |
| 4. Evaluation Synthesizer (per-resource → control-level) |
| 5. Control State Resolver                           |
| 6. Finding Generation & Updates                     |
| 7. Workflow & EventBridge Automation                 |
+-----+
```

Each of these stages is deeply interconnected. The pipeline ensures that Security Hub can perform millions of evaluations across thousands of accounts and regions with high reliability and deterministic behavior.

---

## 3 — Stage 1: Trigger Layer — How Evaluations Begin

---

The Trigger Layer initiates the evaluation process using four categories of triggers:

### 3.1 — Configuration-change triggers (via AWS Config)

The moment configuration of a relevant resource changes—such as an S3 bucket policy, IAM role trust policy, EC2 network interface state, or CloudTrail settings—AWS Config:

- Records the configuration change
- Applies the relevant **Config rule** (if linked to a control)
- Sends the evaluation result to Security Hub

This enables **real-time compliance drift detection**.

---

### 3.2 — Service-detection triggers (GuardDuty, Inspector, Access Analyzer, Macie)

For detection-based controls, these services send findings to Security Hub. These signals trigger Security Hub to:

- Re-evaluate related compliance controls
- Re-assess risk level of related resources
- Update finding severity or confidence if needed

This is critical for standards that require “service X must be enabled” or “all resources must be monitored by service Y.”

---

### 3.3 — Scheduled periodic re-evaluation (“consistency waves”)

Security Hub periodically initiates evaluations to avoid stale states.

- Config-based controls: re-run based on Config evaluation schedule or forcing schedule on drift
- Native controls: scheduled internal re-checks
- Service-integration controls: periodic validation based on external service configuration

These scheduled waves ensure that even long-lived resources remain continuously validated.

---

### 3.4 — On-demand evaluations (manual refresh)

Analysts can click “Re-run control” or “Re-evaluate standard” in the console, which triggers Security Hub’s internal refresh mechanism.

This is used heavily during remediation cycles to validate that a fix now passes.

---

## 4 — Stage 2: Resource Enumeration — Deciding What Must Be Checked

---

For each control, Security Hub must determine which resources are eligible for evaluation.

### 4.1 — Control's resource scope definition

Each control declares:

- Which **resource types** it applies to
- Which conditions define eligibility
- How to map from AWS environment into a resource list

Example:

Control "S3.1 – Public Access Block must be enabled on S3 buckets" applies to all AWS S3 buckets.

### 4.2 — Region-specific enumeration

Security Hub enumerates resources **per region**.

- Some resource types are global (e.g., IAM), but controls run per region
- Enumeration logic adapts depending on whether the resource is truly regional

### 4.3 — Account-bound enumeration

In multi-account environments:

- Each **member account** enumerates its own resources
- The admin/aggregator account aggregates the results
- Security Hub never enumerates resources in other accounts directly

This preserves blast-radius isolation and respects IAM boundaries.

---

## 5 — Stage 3: Underlying Check Execution — Running the Actual Control Logic

---

Security Hub supports **three classes** of evaluation mechanisms:

### 5.1 — Config-based evaluation (most common)

Security Hub leverages AWS Config managed rules.

- Config performs evaluation when configuration changes or at scheduled intervals
- Security Hub simply ingests COMPLIANT / NON\_COMPLIANT results
- This method ensures extremely accurate configuration drift detection

This is the backbone of most CIS and AWS FSBP controls.

---

## 5.2 — Native Security Hub evaluators

Some controls require logic that Config cannot perform.

Examples:

- Checking if Security Hub is enabled in all regions
- Checking GuardDuty or Inspector integration states
- Cross-region or global checks
- Best practices not tied to specific resource types

Security Hub's internal evaluators use AWS APIs or internal metadata curated across the account.

---

## 5.3 — Integration-based evaluation

Some controls rely on signals from other security services:

- GuardDuty must be enabled
- Inspector must have coverage
- IAM Access Analyzer must be enabled

These evaluations depend on the configuration of other services, not on Config or native logic.

---

# 6 — Stage 4: Evaluation Synthesizer — Aggregating Per-Resource Results

Security Hub must turn many per-resource evaluation results into a single control state.

## 6.1 — Per-resource evaluation flow

```
Resource R1 → compliant
Resource R2 → non-compliant
Resource R3 → compliant
Resource R4 → non-compliant
```

The synthesizer aggregates these into:

- List of FAILED resources
- List of PASSED resources
- Partial evaluation notes (e.g., for resources that are ineligible or missing data)

## 6.2 — Control-level aggregation

Control state resolver rules:

- If **any** resource is non-compliant → control is FAILED
- If **all** resources are compliant → control is PASSED

- If **some resources cannot be evaluated** → control becomes WARNING or NOT\_AVAILABLE
- If the control has **no applicable resources**, it may be PASSED (conditions met by default) or NOT\_AVAILABLE depending on standard definitions

This logic must be deterministic and consistent across all standards.

---

## 7 — Stage 5: Control State Resolver — Translating Raw Results into Final Status

---

Security Hub uses a complex rule engine to determine final control state.

### 7.1 — Primary states

- PASSED
- FAILED
- WARNING
- NOT\_AVAILABLE

### 7.2 — Decision Matrix (simplified)

```
If any NON_COMPLIANT resources → FAILED
If all COMPLIANT → PASSED
If some evaluations missing → WARNING
If control cannot run → NOT_AVAILABLE
```

### 7.3 — Additional enhancements

The engine may also consider:

- API throttles
- IAM permission failures
- Service-unavailable errors
- Temporary evaluation inconsistencies
- Internal retries

Final control status reflects not only correctness but **confidence in correctness**.

---

## 8 — Stage 6: Finding Generation and Update Pipeline

---

Once control state is finalized:

## 8.1 — If control FAILED

Security Hub generates or updates **compliance findings**, one per resource or per evaluation depending on the control design.

- Adds detailed resource attributes
- Embeds Compliance block
- Sets Severity (usually LOW/MEDIUM/HIGH depending on the control)
- Sets/Updates timestamps
- Ensures unique finding identity based on generator ID and resource ID

## 8.2 — If control transitions PASSED → FAILED or FAILED → PASSED

Security Hub updates:

- ASFF findings
- Workflow state (sometimes reset or maintained based on user logic)
- UpdatedAt timestamps
- Compliance status in the finding
- Internal dashboards
- EventBridge events

## 8.3 — If control is disabled

Existing findings may stay archived or resolved, and future evaluations do not run.

---

# 9 — Stage 7: Workflow Updates and EventBridge Emission

---

Security Hub emits events for:

- New findings
- Updated findings
- Control state changes
- Workflow changes
- Compliance failures

## 9.1 — Types of emitted events

EventBridge events follow schemas like:

- `Security Hub Findings - Imported`
- `Security Hub Findings - Updated`
- `Security Hub Insight Results`
- `Security Hub Standards Control Evaluated`

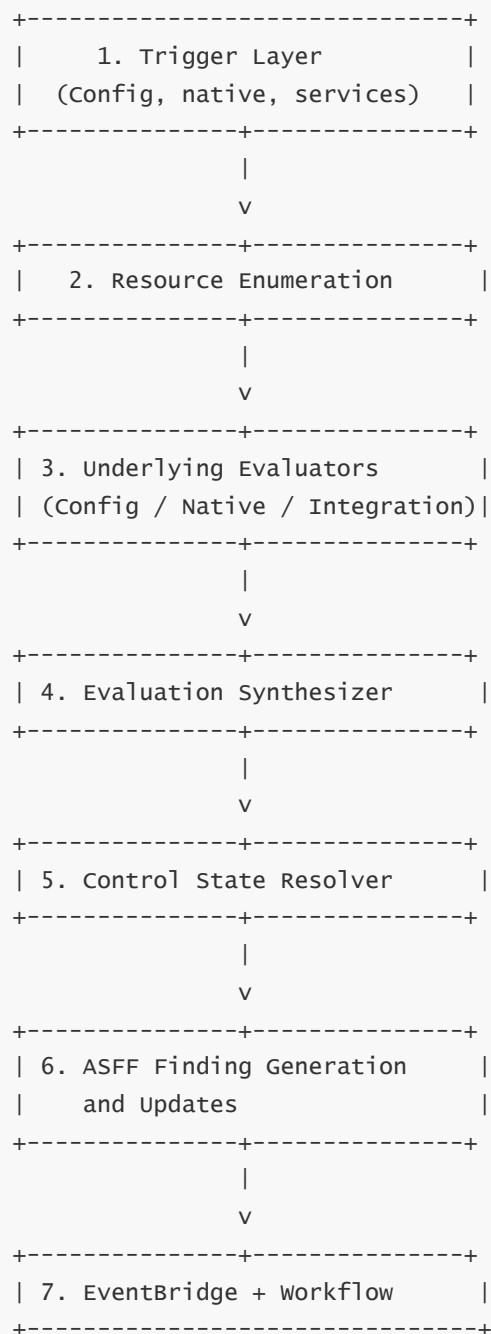


## 9.2 — Uses

These events are used for:

- Automated remediation
- SOAR systems
- SIEM ingestion
- Notification pipelines
- Ticketing systems
- Governance dashboards

## 10 — Full Architectural Diagram of the Evaluation Flow



This shows the continuous, deterministic, multi-stage pipeline that Security Hub executes for each control across all accounts and regions.

---

## 11 — Why This Pipeline Matters: The Internal Philosophy of Security Hub

---

Security Hub is built around one principle:

**“Security state must always reflect reality, not outdated snapshots.”**

This is why the service uses:

- Hybrid triggers
- Partial evaluations
- Dependency-driven updates
- Continuous scoring
- Real-time aggregation
- Regionally isolated processing

This pipeline ensures that the moment something in your infrastructure or threat landscape changes, your compliance and risk posture updates to match it.

---

## 5 — Integration with AWS Services: EventBridge, Config, GuardDuty, Inspector, IAM Access Analyzer

---

### 1 — The Core Philosophy Behind Service Integrations in Security Hub

---

Security Hub’s architecture is intentionally designed to **not reinvent specialized security engines**. Instead, Security Hub acts as the **central security aggregation and normalization layer** for the entire AWS ecosystem.

This means the integrations with other AWS security services are not superficial or “notifications-only” integrations—they are **deep functional pipelines** that:

- Transport findings
- Transport configuration evaluations
- Transport threat signals
- Transport identity risk signals
- Transport vulnerability and malware intelligence

- Trigger compliance evaluations
- Trigger workflow updates
- Trigger EventBridge automation
- Feed the internal ASFF normalization engine
- Feed cross-account aggregation
- Feed control evaluations for standards
- Feed insights, dashboards, and risk models

The integration is not “Security Hub reading events shipped from other services,” but instead:

**“Other AWS services feed raw or semi-processed intelligence into Security Hub, and Security Hub turns it into a unified, structured, queryable, automatable security state.”**

To understand this, we must break down each major integration type.

---

## 2 — Integration with EventBridge: Security Hub’s Nervous System

---

EventBridge is the **primary event transport and orchestration bus** for Security Hub. It plays three roles:

### 2.1 — Ingestion Path for Findings

Many AWS services emit events to EventBridge, which Security Hub consumes as:

- New findings
- Updated findings
- Resource configuration change notifications
- Compliance signals

For example, if Inspector detects a new CVE on an EC2 instance, Inspector emits a finding event → EventBridge → Security Hub’s ingestion layer → normalization engine.

---

### 2.2 — Outbound Event Stream for Automation

When Security Hub receives or updates a finding, it emits **Security Hub Findings - Imported** or **- Updated** events to EventBridge.

These can be used to trigger:

- Lambda functions
- SSM Automation documents
- Step Functions workflows
- Ticketing system integrations
- SOAR runbooks (e.g., Splunk Phantom, Demisto, Cortex XSOAR)

This makes EventBridge the **automation backbone** for response and remediation.

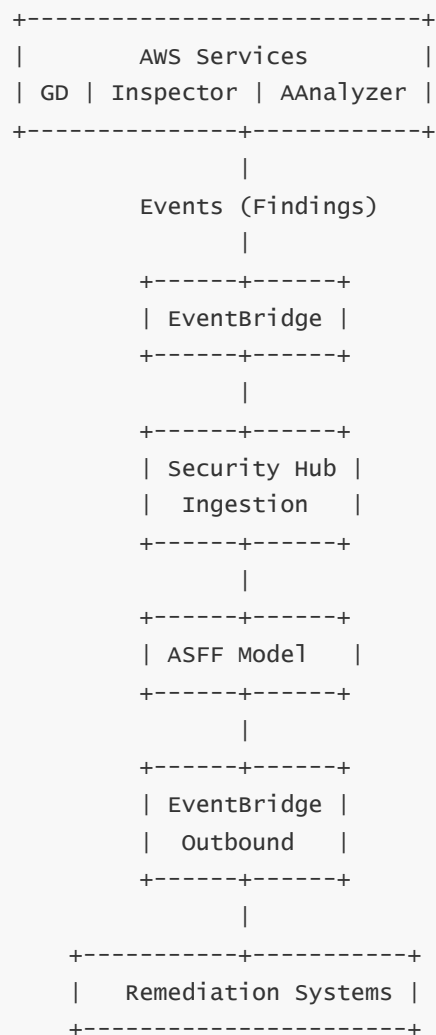
---

## 2.3 — Compliance and Control State Events

Security Hub also emits “Standards Control Evaluated” events, which allow organizations to:

- Automate remediation whenever a control fails
- Enforce security guardrails
- Send notifications
- Enrich SIEM dashboards
- Synchronize compliance posture with GRC tools

## 2.4 — EventBridge Integration Diagram



EventBridge is the **bi-directional message bus** that connects Security Hub with the rest of AWS.

## 3 — Integration with AWS Config: The Compliance Backbone

---

AWS Config is the **core compliance engine** that Security Hub uses for the majority of its best-practice assessments.

### 3.1 — Config as the Configuration Source of Truth

Config tracks every configuration change in your environment. Security Hub uses these configuration streams to evaluate:

- Resource drift
- Misconfiguration
- Cloud security posture
- Regulatory compliance
- AWS Best Practice compliance

### 3.2 — Config Rules as Control Implementations

Most Security Hub controls are implemented as:

- **AWS Config Managed Rules** (prebuilt by AWS)
- **Custom Config Rules** (user-defined, but used by Security Hub)

These Config rules evaluate:

- VPC configurations
- IAM configurations
- S3 configurations
- EC2 networking
- Encryption settings
- Logging settings
- Security group exposure
- Public accessibility
- RDS configurations
- CloudTrail settings
- KMS key rotation

Security Hub uses Config's evaluation results to power its compliance engine.

---

### 3.3 — Config Evaluation → Security Hub Control State

Flow:

```
Resource Change → Config → Config Rule Evaluation → Evaluation Result →  
Security Hub Control State Resolver → Compliance Status → Finding Generation
```

Config drives compliance with extremely high accuracy because it evaluates on every configuration change, not just on schedules.

### 3.4 — Why Config is Critical

Without Config, Security Hub would lose:

- Real-time configuration visibility
- Resource inventory
- Drift detection
- Compliance mapping
- Change-triggered evaluations

Security Hub relies on Config as its **primary compliance computation engine**.

## 4 — Integration with GuardDuty: Threat Detection → Normalized Findings

GuardDuty feeds **behavioral threat detections** into Security Hub. These include:

- Compromised EC2 instances
- Compromised IAM role usage
- Credential exfiltration
- Reconnaissance
- Crypto mining
- DNS exfiltration
- Malware behavior
- API anomaly detections

### 4.1 — GuardDuty → Security Hub Flow

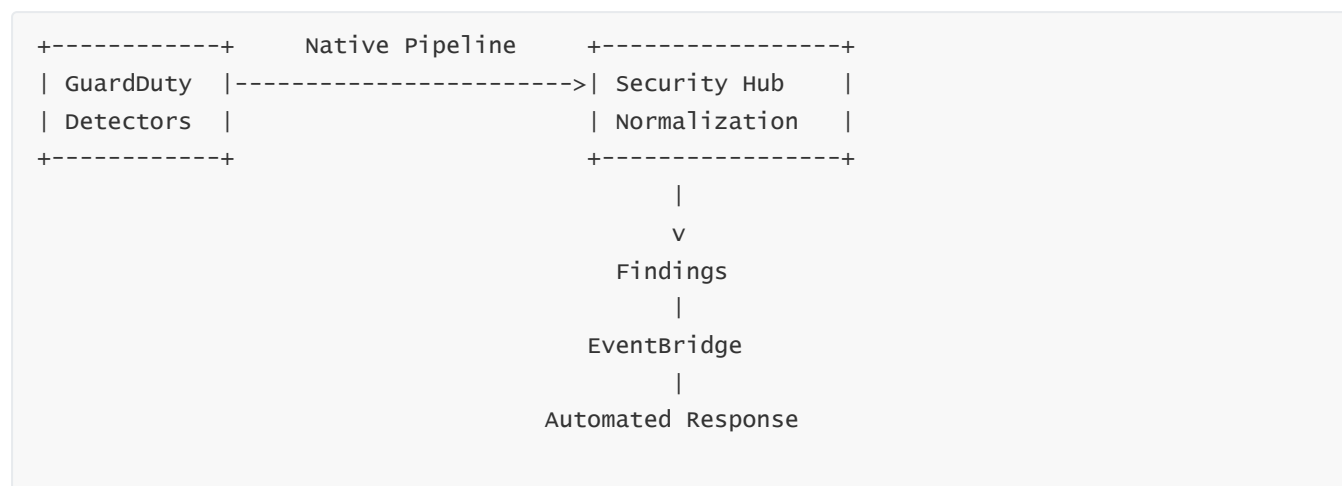
```
Threat Activity → GuardDuty Detector → GD Finding Generated →  
GD → Security Hub Ingestion → ASFF Normalization →  
Risk Scoring → Insights → Dashboards → Automation Triggers
```

## 4.2 — Why GuardDuty Integration is Deep

Security Hub doesn't just display GuardDuty findings. It:

- Normalizes severity
- Adds workflow state
- Assigns compliance mapping (if relevant)
- Integrates into multi-account aggregator
- Enables cross-service correlation (e.g., link with Config violations)
- Powers threat insights (e.g., "Top 10 exploited resources")

## 4.3 — Combined Architecture: GD + SH



Security Hub becomes the **central threat visibility console** for all GuardDuty activity.

## 5 — Integration with Amazon Inspector: Vulnerabilities, CVEs, Malware, Packages

Inspector is AWS's **vulnerability management engine**. It detects:

- CVEs in EC2, Lambda, and container images
- Malware
- Unpatched packages
- Exploitability indicators
- Package version drift

### 5.1 — Inspector → Security Hub Flow

Package Scan → CVE Detected → Inspector Finding →  
Inspector → Security Hub → ASFF Normalization →  
Insights → Trending → Cross-Account Aggregation → Automation

Inspector sends extremely rich metadata, which Security Hub stores in the `vulnerabilities[]` block.

## 5.2 — Data Included from Inspector in Security Hub

Security Hub stores:

- CVE metadata
- CVSS base score, vectors, attack complexity
- Affected packages and versions
- Fix versions
- Layer information for container images
- Runtime behavior (for malware)

This makes Security Hub the central repository for vulnerability management across all accounts.

---

## 6 — Integration with IAM Access Analyzer: Identity Risk & Public Exposure Detection

Access Analyzer evaluates:

- Resource sharing
- Public exposure
- Cross-account access
- External principal access
- Unintended trust relationships

### 6.1 — IAM Analyzer → Security Hub

Analyzer Policy Check → External Access Found →  
Access Analyzer → Security Hub → ASFF Finding →  
Resource Attribution → Compliance Mapping

### 6.2 — Why IAM Analyzer Integration Matters

IAM access exposure issues often overlap with Config or GuardDuty signals. Security Hub is the place where:

- Exposure findings
  - Misconfig findings
  - Threat findings
- are all brought together.

Example correlation:

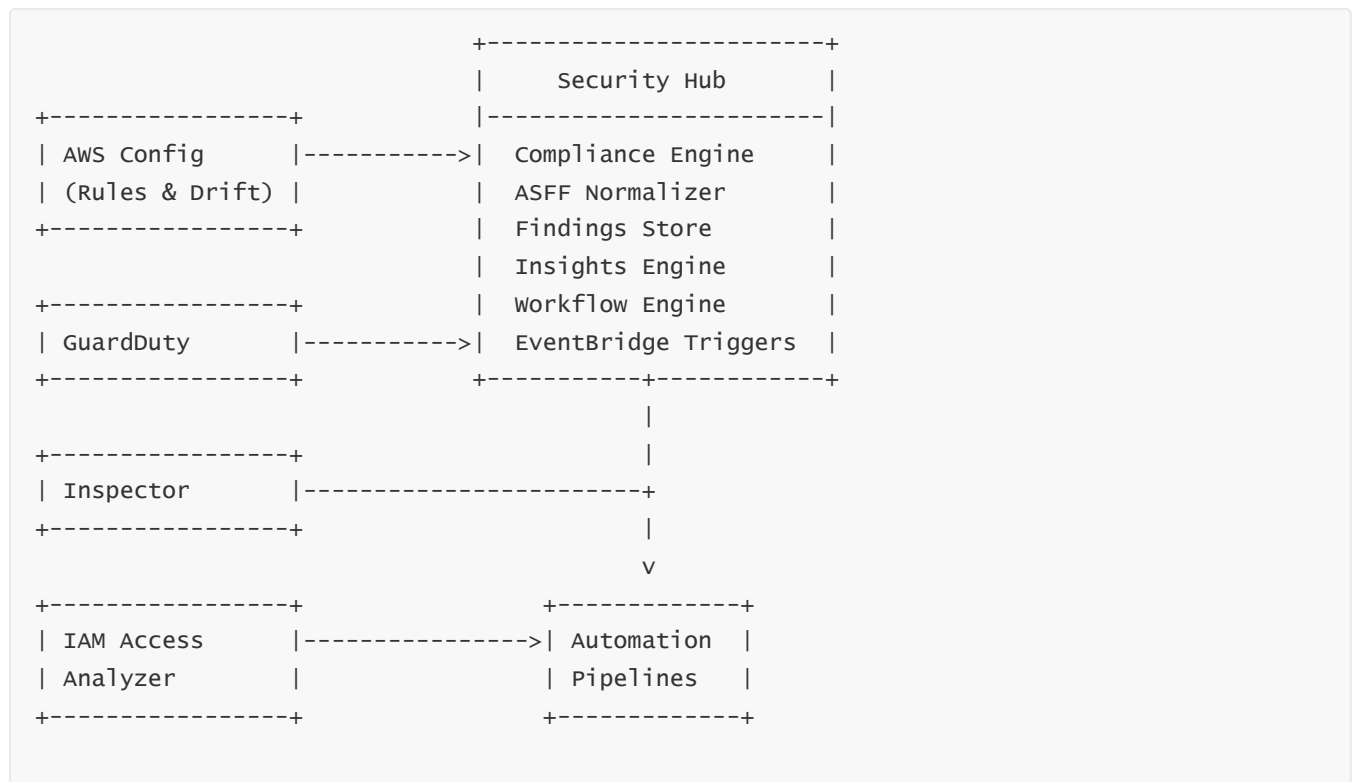
- Access Analyzer: "Role allows external principal."
- GuardDuty: "Role used anomalously."



- Config: “IAM role lacks MFA enforcement.”

Security Hub is the only service where these signals merge into a single unified risk picture.

## 7 — Combined Multi-Service Integration Diagram



This diagram reflects the *true internal structure*—Security Hub sits at the center, ingesting, normalizing, analyzing, correlating, and automating across all AWS security signals.

## 8 — Why Security Hub’s Integration Model is So Powerful

Security Hub becomes the **single pane of glass** because it:

- Normalizes threat, compliance, configuration, and identity signals
- Aggregates across accounts and regions
- Provides unified scoring and insights
- Allows automation from a single central event source
- Creates ASFF records that external SIEMs and SOARs can ingest cleanly
- Maps signals to compliance frameworks
- Maintains workflow state for every finding
- Coordinates multi-service, multi-signal correlation

No single AWS security service alone can do this.

Security Hub is the **security brain** of the AWS ecosystem.

# 6 — How Security Hub Works with Partner Tools and External Security Solutions

---

## 1 — Philosophy of Third-Party Integration: Security Hub as the AWS Security “Data Spine”

---

Security Hub was designed to be the **central integration plane** for AWS native services *and* external security ecosystems.

Unlike other cloud platforms where integrations are bolt-ons, AWS designed Security Hub so that *partner tools are first-class contributors* to the overall security posture.

The architectural philosophy is:

- **Any external security system** (cloud posture tool, vulnerability scanner, endpoint platform, SIEM, CSPM, CWPP, container scanner, SAST/DAST tool, compliance scanner, etc.)
- Can **push intelligence into Security Hub**
- Using the **AWS Security Finding Format (ASFF)**
- So the data becomes **normalized, queryable, aggregatable**, and **actionable**
- Just like AWS native findings

In other words:

**Partner findings become indistinguishable from native AWS security findings.**

The same dashboards, insights, automation, workflow, compliance mapping, and cross-account aggregation apply to them.

This transforms Security Hub into an **“operational security backbone”** where cloud, network, identity, vulnerability, and compliance signals from many vendors converge.

---

## 2 — Partner Categories that Integrate with Security Hub

---

Security Hub supports integration with over 75+ partner solutions (the number grows continuously). These belong to categories such as:

- Cloud Security Posture Management (CSPM)
- Endpoint Detection & Response (EDR)
- Cloud Workload Protection (CWPP)
- SIEM / SOAR
- Penetration testing tools
- Vulnerability scanners
- Threat intelligence platforms
- Data security platforms
- Network monitoring tools

- Compliance management products
- API security platforms
- Runtime container security

These tools integrate using the same ASFF format as GuardDuty, Inspector, and others.

---

## 3 — The Partner Integration Pipeline: How Third-Party Findings Enter Security Hub

---

### 3.1 — Core Requirement: Findings Must Be in ASFF Format

To ingest third-party data, a partner solution must:

- Produce findings that conform to the **AWS Security Finding Format (ASFF)**
- Use Security Hub's **BatchImportFindings** API
- Provide correct resource modeling, severity, timestamps, workflow mapping, and product metadata

ASFF makes all external findings behave like AWS native ones.

---

### 3.2 — The Full Ingestion Flow

```
External Scanner / Tool
  |
  | Generates raw security event
  v
Partner Integration Module (ASFF transformer)
  |
  | Converts event → ASFF-compliant finding
  v
Security Hub BatchImportFindings API
  |
  | Security Hub Ingestion Pipeline
  v
ASFF Normalization Engine (validates, canonicalizes)
  |
  v
Findings Store (regional)
  |
  v
Insights Engine / Dashboards / EventBridge / Aggregation
```

Where native AWS services “skip” the ASFF transformer because they already produce ASFF internally, external partners must transform their own formats.

---

## 4 — What Security Hub Does After Ingesting Partner Findings

---

Security Hub treats partner findings **identically** to AWS findings. After ingestion, Security Hub performs:

### 4.1 — Field Validation & Normalization

Security Hub ensures the finding is:

- Valid JSON
- Fields conform to schema
- Severity valid
- Resources correctly modeled
- ProductArn unique
- Timestamps canonical
- Workflow consistent

Incorrect findings are rejected with errors.

---

### 4.2 — Deduplication & Update Logic

Security Hub uses `Id` + `ProductArn` as the unique key.

If the partner tool sends the same issue again, Security Hub **updates** rather than duplicates.

---

### 4.3 — Multi-Account / Multi-Region Aggregation

Partner finding for account A in region R:

- Ingested into that account/region
  - Automatically aggregated into the delegated admin account
  - Rolled up into org-level insights
- 

### 4.4 — Automation Triggering

Security Hub emits “Findings Imported” or “Findings Updated” events to EventBridge.

From here:

- Lambda/SSM can run remediation
- SIEM/SOAR pipelines can execute playbooks
- Ticketing systems can sync the event
- Notifications can be sent
- GRC tools can map to compliance

Security Hub ensures partner findings can be used in exactly the same workflows.

---

## 4.5 — Dashboard & Insight Inclusion

Once normalized, partner findings appear in:

- Insights
- Aggregated severity dashboards
- Resource-centric views
- Multi-service correlation pages

This enables unified security posture visibility.

---

## 5 — Partner → Security Hub ProductArn Model

Every partner product integrating with Security Hub must define a ProductArn:

```
arn:aws:securityhub:<region>::product/<vendor>/<product>
```

This ARN allows Security Hub to:

- Attribute findings to that vendor
- Filter results
- Apply queries and insights
- Support cross-region isolation
- Manage updates and lifecycle

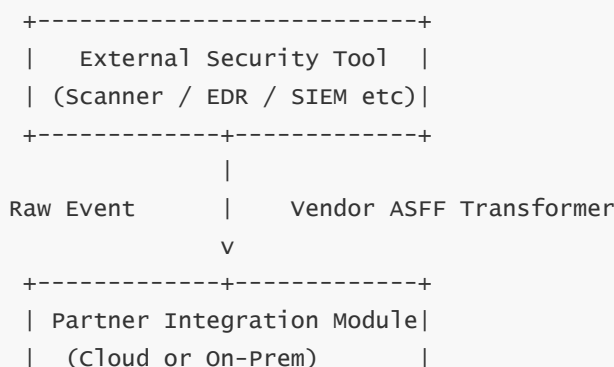
For example, a CSPM tool like Prisma Cloud might have:

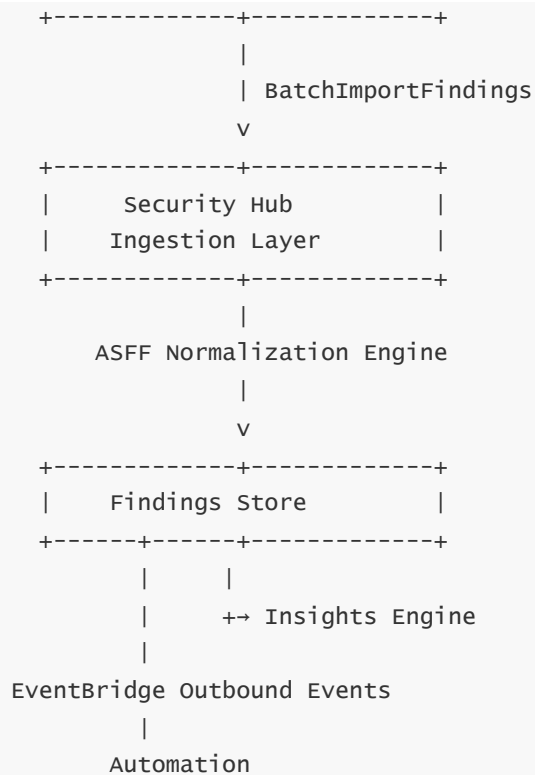
```
arn:aws:securityhub:us-east-1::product/paloaltonetworks/prismacloud
```

Security Hub uses this ARN to logically separate provider namespaces.

---

## 6 — Deep Architecture Diagram of Partner Integration into Security Hub





This reflects the actual flow used by all partner tools.

## 7 — Types of Integration Models

### 7.1 — Direct Partner Integration (Cloud-to-Cloud)

Partner SaaS calls Security Hub's API directly.

Examples:

- Lacework
- Palo Alto Prisma Cloud
- Wiz
- Orca
- Rapid7 InsightVM
- Trend Micro Vision One

### 7.2 — Connector-Based Integration (Deployed in Customer Account)

Customer deploys a connector (Lambda, ECS, EC2 agent) which:

- Collects data from a scanner
- Converts to ASFF
- Pushes to Security Hub

Examples:

- Qualys
- Tenable
- Burp Suite automation connector

## 7.3 — SIEM/SOAR Integrations

SIEM or SOAR fetches or pushes findings:

- Splunk
- QRadar
- Sumo Logic
- Datadog
- Panther
- Cortex XSOAR

These tools can push custom findings (e.g., threat intel matches) or pull Security Hub findings.

---

## 8 — What Makes ASFF So Critical for Partner Integrations

ASFF gives partner findings:

- **Standardized severity**
- **Standardized resource representation**
- **Unified remediations**
- **Unified workflow state**
- **Unified timestamps**
- **Unified compliance mapping**
- **Unified automation triggers**
- **Unified cross-account aggregation**

This eliminates the historical problem of “every security tool uses a different format,” making automation nearly impossible.

ASFF transforms the AWS security ecosystem into a **clean, structured, correlation-friendly environment**.

---

## 9 — How Partner Findings Influence Compliance and Posture

Partner findings can influence:

- Security posture insights
- Organizational dashboards
- Risk scoring
- Compliance mapping if the product provides mappings

- Cross-account posture summaries
- Resource-centric problem classification

Example:

A partner CSPM tool detects “S3 bucket open to Internet.”

Security Hub normalizes it → reflects in S3 risk insights → contributes to account-level high severity counts → shows in dashboards.

Security Hub does **not** evaluate partner controls for compliance standards unless the partner maps them, but the findings still enhance posture.

---

## 10 — Multi-Account Partner Integration Behavior

### 10.1 — Partner pushes findings into each member account

The partner must have access to each member account (through delegated roles or connectors).

### 10.2 — Security Hub aggregates these

Security Hub automatically centralizes findings in the **delegated admin** account.

### 10.3 — Organization-wide dashboards

Admins see unified partner data across the entire org.

### 10.4 — Workflow syncing

Workflow state changes in the admin account can propagate back to the member account findings.

---

## 11 — Using Partner Findings for Automated Remediation

Because partner findings enter Security Hub as ASFF events, EventBridge rules can be written like:

```
If ProductArn = partner/foo
And Severity.Label = HIGH
And Resource.Type = AwsEc2Instance
→ Run remediation Lambda
```

This allows:

- Automated quarantine
- Automated blocking policies
- Automated IAM corrections
- Automated notifications to SOC
- Automated ticket creation



Partner findings become a **first-class automation trigger**.

---

## 12 — Bi-Directional Integrations: Security Hub → SIEM/SOAR → Back to Security Hub

---

Some partners support a two-way model:

- Findings pushed to Security Hub
- Workflows modified in Security Hub
- Status synced back to the partner platform
- Tickets/alerts auto-closed in SIEM or SOAR based on SH updates

This allows Security Hub to serve as the **central source of truth** for finding lifecycle.

---

## 13 — Why Partner Integration Architecture Makes Security Hub Uniquely Powerful

---

Security Hub becomes:

- The **central risk repository**
- The **real-time security posture dashboard**
- The **correlation layer across cloud, identity, vulnerability, threat, and configuration**
- The **cross-account unifier**
- The **automation trigger center**
- The **translation layer for external tools**

AWS native tools + partner tools + internal custom tools all converge into one consistent model.

No other cloud provider offers this degree of **first-class standardization + correlation + multi-account visibility** with external security ecosystems.

---

## 7 — Cross-Account and Cross-Region Aggregation Architecture in AWS Security Hub

---

### 1 — Why Aggregation Exists: Security Hub as a Multi-Account Governance Platform

---

AWS environments rarely contain a single account. Real-world architectures involve dozens, hundreds, or even thousands of AWS accounts—segmented by:

- Organizational Units (Prod / Dev / Security / Sandbox)

- Departmental or business boundaries
- Environment boundaries
- Workload separation
- Isolation and least privilege requirements

Security Hub was designed from the ground up to operate as a **federated, hierarchical, multi-account security management plane**.

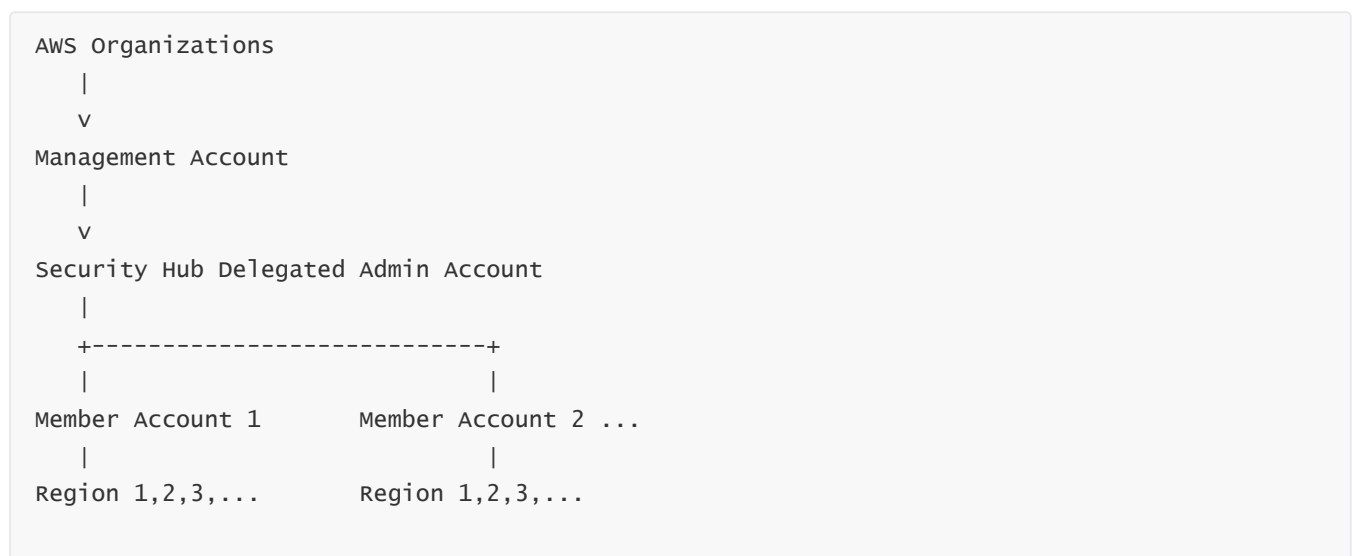
Aggregation enables Security Hub to:

- Collect findings from all accounts
- Collect compliance statuses from all accounts
- Consolidate them in a **single delegated admin** account
- Allow security teams to see organization-wide posture
- Enforce consistent standards from the top
- Enable central automation
- Maintain isolation between member accounts
- Preserve regional control boundaries

This architecture turns Security Hub into the **global governance dashboard** for the entire AWS footprint.

## 2 — The Administrative Hierarchy: Management Account → Delegated Admin → Member Accounts

AWS Organizations defines the top-level governance structure, and Security Hub integrates natively into it.



### 2.1 — Management Account

- Enables Security Hub integration with the Organization
- Assigns **one** account as the **delegated admin**

## 2.2 — Delegated Admin Account

This is the central brain for Security Hub.

It:

- Receives aggregated findings from all member accounts
- Receives aggregated compliance results
- Shows unified organization security posture
- Allows org-level remediation and automation
- Propagates standards across accounts

## 2.3 — Member Accounts

Each member account:

- Runs its own Security Hub instance per region
- Ingests its own findings
- Evaluates its own compliance controls
- Sends results to the delegated admin

Member accounts retain ownership of resources, but the admin owns **visibility and governance**.

---

# 3 — The Regional Isolation Principle: Aggregation Occurs per Region

---

Security Hub is a **regional service**, and aggregation never breaks AWS's region-isolation boundary.

## 3.1 — Findings stay in the region where they occur

If GuardDuty in `us-east-1` generates a finding, that finding stays in:

- `us-east-1` for storage
- `us-east-1` for evaluation
- `us-east-1` for organization-level aggregation

## 3.2 — Aggregator pulls from all accounts *within the same region*

Delegated admin in `us-east-1` gets findings from all member accounts in `us-east-1`, but **not** from `us-west-2`.

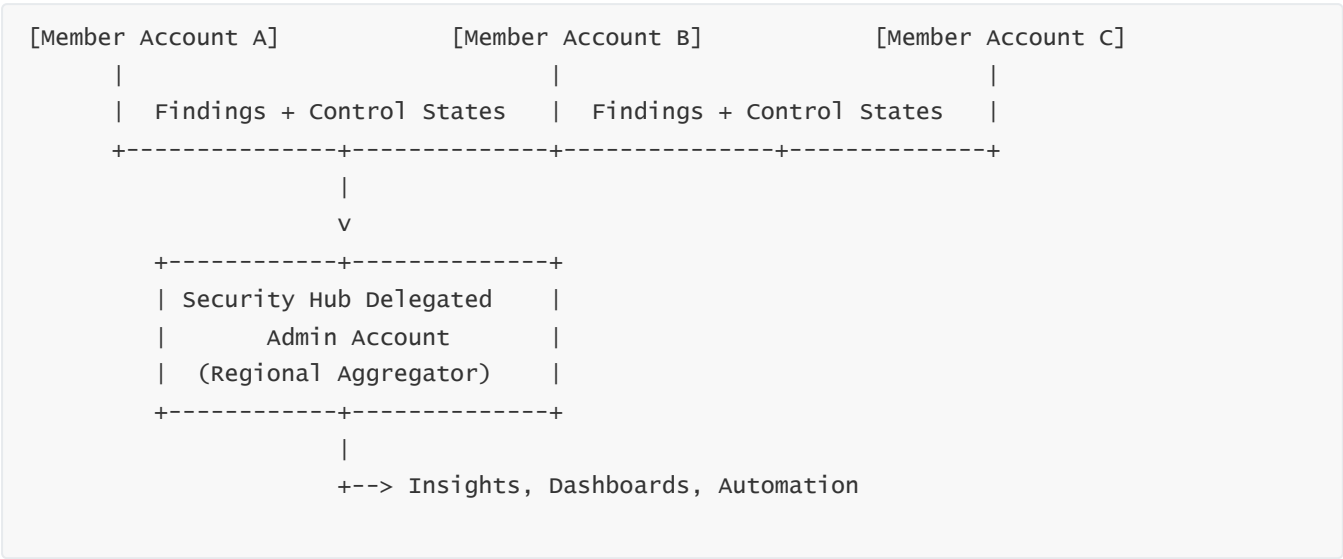
### 3.3 — Multi-region architecture requires admin to enable Security Hub in all regions

To get a true global view, the delegated admin must enable SH in:

- us-east-1
- us-west-1
- eu-west-1
- ap-south-1
- ...and so on.

This maintains data sovereignty and compliance boundaries.

## 4 — Deep Internal Aggregation Flow: How Data Moves Across Accounts



Breakdown of the flow:

### 4.1 — Member account generates findings

From:

- GuardDuty
- Inspector
- Config rules
- Security Hub native controls
- IAM Access Analyzer
- Partner tools

## 4.2 — Member account sends findings to admin

Security Hub uses internal AWS service-to-service communication to replicate findings.

## 4.3 — Admin stores aggregated findings

Admin maintains:

- Per-account posture
- Per-region posture
- Per-standard posture
- Combined resource insight views

## 4.4 — Admin provides org-wide dashboards

With aggregated data, the admin account shows:

- Top failing controls across all accounts
- Accounts with highest number of high-severity findings
- Compliance percentage across all accounts
- Organization-wide risk patterns
- Account drift and anomalies

---

# 5 — The Aggregation Data Pipeline Internals

Security Hub uses a **two-phase replication job** internally:

## 5.1 — “Push” phase (from member)

Member accounts push:

- Findings
- Updates
- Workflow changes
- Compliance evaluation results

into the admin’s data plane.

---

## 5.2 — “Consolidation” phase (in admin)

The admin merges:

- Duplicate findings
- Updates from child accounts
- Workflow status overrides
- Compliance scores

To create a unified **Organization ASFF Dataset**:

```
Organization Security Dataset
├─ Account A
│   └─ Findings
│       └─ Controls
├─ Account B
├─ Account C
└─ ...
```

This dataset powers org-wide insights.

---

## 6 — Workflow Synchronization Across Accounts

When a finding's **workflow status** is changed in the admin account (e.g., NEW → NOTIFIED → RESOLVED → SUPPRESSED), Security Hub can push the workflow state back to member accounts.

### 6.1 — This ensures:

- Central SOC decisions propagate globally
- Investigations are synchronized
- No inconsistent states exist
- Cross-account automation becomes predictable

### 6.2 — But note

Member modifications may override admin changes depending on workflow permissions.

---

## 7 — Compliance Aggregation Architecture

Compliance is aggregated separately from findings:

```
Member Account → Control Results → Admin Account → Standard Summary
```

### 7.1 — Member performs evaluations

Controls evaluated locally via:

- Config
- Native evaluators
- External service checks

## 7.2 — Results pushed to admin

Admin stores control states grouped by:

- Account
- Region
- Standard
- Control

## \*7.3 — Admin computes org-wide standard compliance

Example:

Control “S3.1: No public buckets” might be:

- PASSED in 148 accounts
- FAILED in 3 accounts
- NOT\_AVAILABLE in 1 account

Admin displays:

- Compliance percentages
- Per-account deviations
- Trend lines
- Worst-affected areas

---

## 8 — Cross-Region Aggregation Architecture: Multi-Region Admin View

Security Hub does **not** replicate data automatically across regions.

To get a global posture, the admin must enable SH in **each region**.

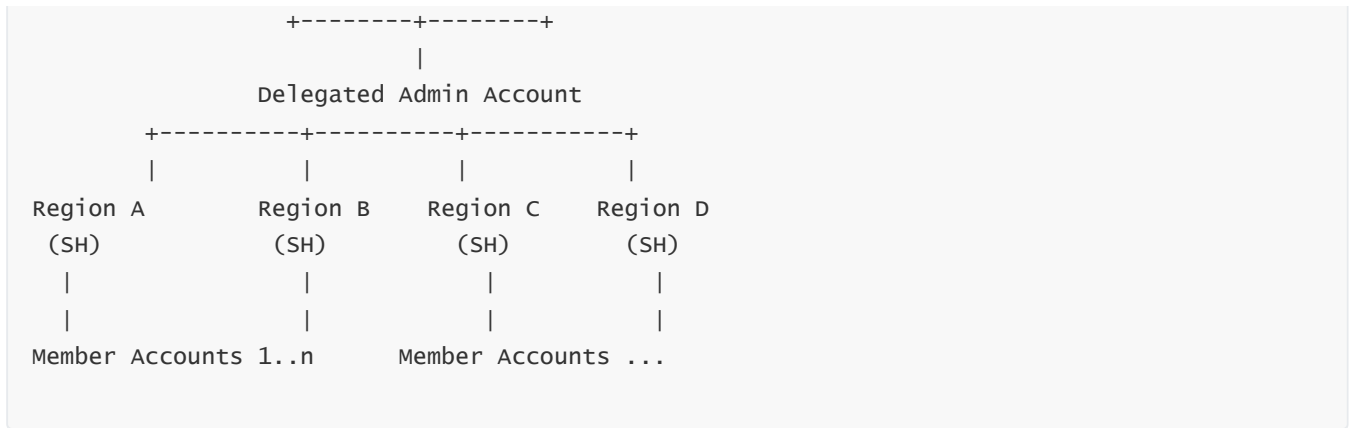
```
Region us-east-1 → Admin us-east-1
Region us-west-2 → Admin us-west-2
Region eu-west-1 → Admin eu-west-1
```

The “global view” is achieved through console federation or reporting tools—not automatic replication.

---

## 9 — Multi-Region, Multi-Account Data Plane Diagram

```
      AWS Organizations
      |
      |
+-----+-----+
| Management Acc. |
```



Each region produces its own aggregated dataset.

The admin consolidates per-region views manually or using SIEM/SOAR/GRC pipelines.

---

## 10 — Benefits of Security Hub’s Aggregation Model

### 10.1 — Centralized Governance

Security, audit, and compliance teams use one pane to monitor all accounts.

### 10.2 — Consistent Standards Enforcement

CIS or FSBP can be enabled across 100s of accounts in a single operation.

### 10.3 — Unified Policies and Controls

Admin ensures standard enablement is uniform and enforced.

### 10.4 — Organization Security Insights

“Top 10 riskiest accounts”

"Top failing controls across org"

“Accounts with GuardDuty disabled”

All derived from aggregated data.

### 10.5 — Automation Across the Entire AWS Footprint

EventBridge in the admin account triggers remediations across accounts.

---

## 11 — Why AWS Chose a Delegated Admin Model Instead of Global Replication

There are strategic reasons:

- **Data sovereignty laws**
- **Regional compliance boundaries**

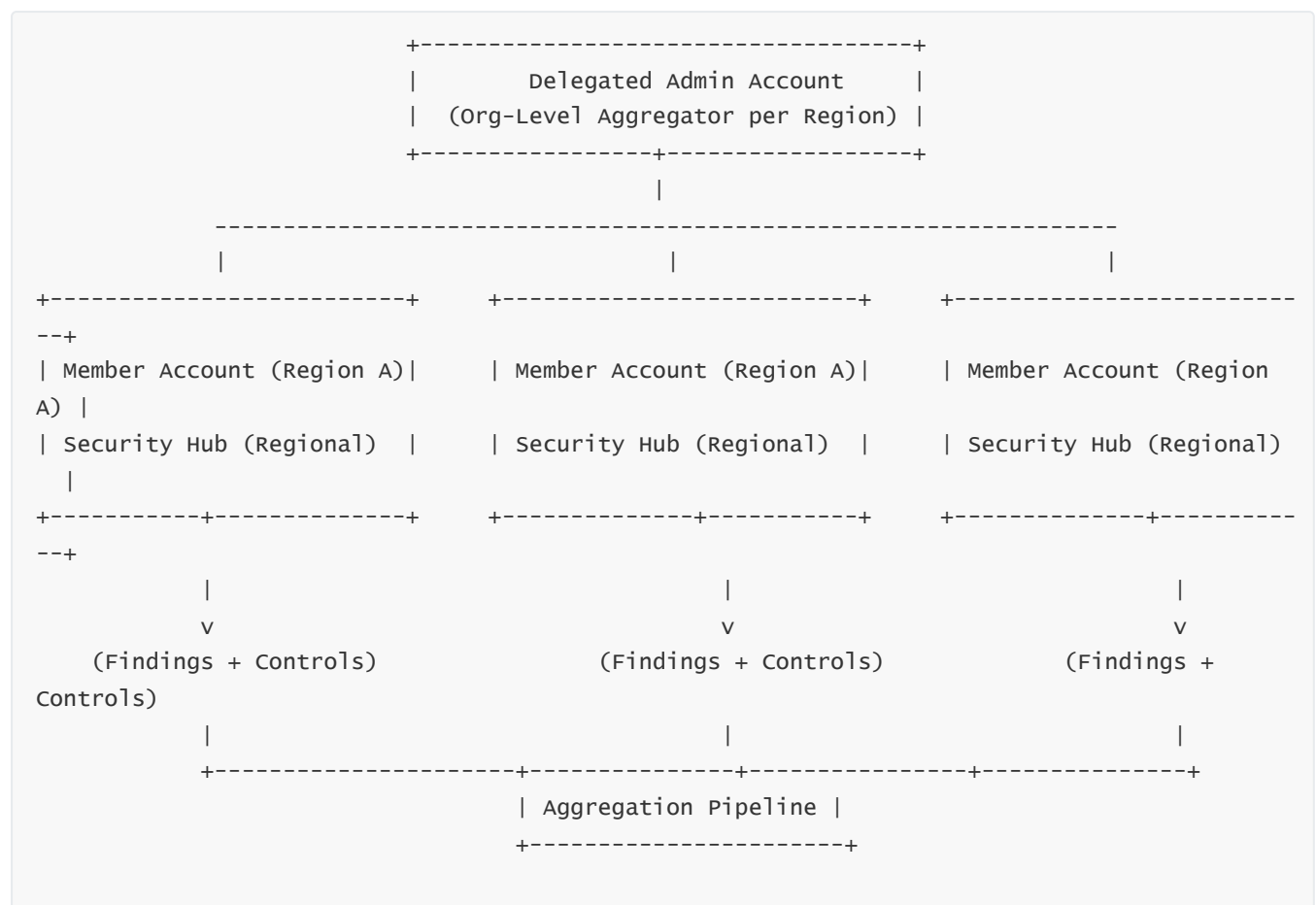


- **Blast-radius reduction**
- **Minimal cross-region data transfer**
- **Predictable performance**
- **Scalable org deployments**
- **Separation of concerns**

The admin account acts as a **governance hub**, but each region-account pair remains a **compute and storage island**.

AWS wants strong **compartmentalization** with optional **central visibility**.

## 12 — Full Multi-Layer Aggregation Diagram



This is what actually powers organization-wide security visibility.

## 13 — Mental Model Summary

Security Hub aggregation is built on three rules:

1. **Every account evaluates its own resources.**
2. **Every region stores findings in that region.**
3. **Delegated admin centrally aggregates per-region, per-account data into governance dashboards.**

This is the **federated, regionally isolated, centrally governed** architecture that AWS uses for cloud-scale security posture management.

---

## 8 — Automation Workflows: EventBridge Rules, Playbooks, and Response Pipelines

---

### 1 — Why Automation is a Core Pillar of Security Hub

---

Security Hub is not designed as a passive dashboard.

Its architecture assumes that security operations must be:

- **Continuous**
- **Automated**
- **Event-driven**
- **Scalable across hundreds of accounts and regions**
- **Integrated with remediation systems, SIEM, SOAR, and ticketing systems**

Automation is therefore not just a feature but a **primary operational mode** of Security Hub.

It transforms Security Hub from “a console that shows security issues” into **a control plane that can trigger response actions across all AWS accounts**.

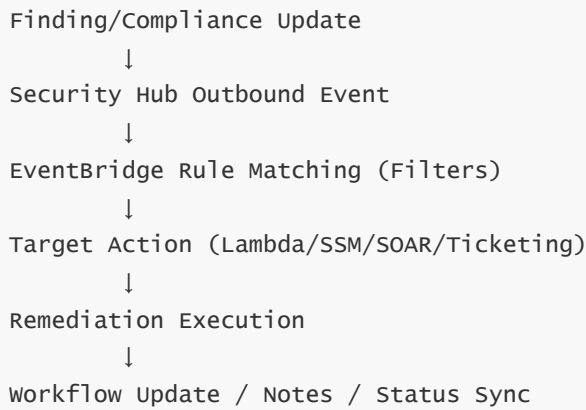
This is achieved through a tightly integrated automation pipeline connecting:

- **EventBridge**
  - **Lambda**
  - **Systems Manager Automation**
  - **Step Functions**
  - **Security Orchestration Platforms (SOAR)**
  - **Vendor response engines**
  - **Custom remediation frameworks**
- 

### 2 — High-Level Automation Flow: Signal → EventBridge → Action

---

The core automation loop is:



Security Hub becomes the **center of the detection-to-response feedback cycle**.

---

## 3 — The Three Types of Automation Events in Security Hub

---

Security Hub emits three primary event types:

### 3.1 — “Security Hub Findings - Imported”

Triggered whenever a new finding arrives.

Used to initiate:

- Alerts
- Immediate triage
- Initial response

### 3.2 — “Security Hub Findings - Updated”

Triggered when:

- Severity changes
- Workflow changes
- Compliance status changes
- Resource details updated
- Remediation results applied

These events are used to synchronize lifecycle with SOAR systems or to trigger secondary automation.

### 3.3 — “Security Hub Standards Control Evaluated”

Triggered when a **compliance control evaluation** completes.

Used for:

- Compliance-driven remediation
- Automatic enforcement measures
- GRC pipelines

- Reporting events
- 

## 4 — EventBridge as the Automation Brain

---

EventBridge is the **pattern-matching and routing engine** that controls automation flows.

### 4.1 — Event Matching Logic

A Security Hub event is matched against EventBridge rules using conditions like:

- Severity
- Compliance status
- ProductArn
- GeneratorId
- Resource.Type
- Workflow status
- Account or region
- EventType (imported/updated)

This allows extremely precise targeting.

Example:

Trigger only when:

- A **High severity** finding
- From **Inspector**
- For a **production account**
- With **active state**
- And **EC2 instance** as resource

This drives highly selective remediation.

---

### 4.2 — EventBridge Routing

EventBridge can route matching events to many targets:

- Lambda
- ECS tasks
- SQS queues
- SNS notifications
- Step Functions state machines
- Event buses in other accounts
- Systems Manager Automation documents
- Kinesis streams

- Partner SOAR integrations

This makes the automation framework **multi-directional** and **multi-system**.

---

## 5 — Lambda as the Remediation Engine

---

Lambda is the most commonly used remediation tool.

### 5.1 — Why Lambda Works Well

- Zero infrastructure management
- Fast execution
- Easy API integration
- Works across all AWS services
- Low operational overhead
- Native IAM integration
- EventBridge → Lambda binding is trivial

### 5.2 — Typical Lambda Remediation Examples

- Remove public access from S3 buckets
- Fix insecure security group rules
- Enable encryption
- Recreate missing CloudTrail trails
- Rotate IAM credentials
- Disable public AMIs
- Stop compromised EC2 instances
- Quarantine workloads

Lambda becomes the “hands” of Security Hub in automation.

---

## 6 — Systems Manager Automation: Enterprise-Grade Remediation

---

SSM Automation provides:

- Multi-step workflows
- Approval steps
- Conditional logic
- Rollback logic
- Parameter passing
- Audit logs
- Permissions control

Security Hub → EventBridge → SSM Automation is used for:

- Complex remediations
  - Organization-wide remediation flows
  - Controlled change processes
  - Infrastructure-level corrections
  - Patching and vulnerabilities
  - Governance-enforced fixes
- 

## 7 — Step Functions: Orchestration of Multi-Step Response Pipelines

---

Step Functions are used when response requires **sequenced, stateful, multi-component** processes.

Example:

1. Extract metadata from finding
2. Notify SOC
3. Invoke remediation Lambda
4. Validate fix
5. Update Security Hub workflow
6. Post-update to ticketing system
7. Archive finding

This makes Step Functions the backbone of **large-scale enterprise remediation pipelines**.

---

## 8 — Integration with SOAR Platforms

---

SOAR systems consume Security Hub events and automate investigations and responses.

Examples include:

- Palo Alto Cortex XSOAR
- Splunk Phantom
- IBM QRadar SOAR
- Rapid7
- Swimlane
- Fortinet SOAR

## Common SOAR workflows:

- Correlate findings across multiple tools
- Run enrichment (IP reputation, malware sandboxing)
- Open incident tickets
- Execute playbooks
- Apply remediation
- Update Security Hub workflow state

SOAR + Security Hub builds a **complete detection-to-resolution** loop.

---

## 9 — Ticketing Integrations: Jira, ServiceNow, Zendesk

Security Hub events can trigger:

- Jira ticket creation
- ServiceNow incident creation
- Ticket updates
- Closure synchronization

Workflow in a nutshell:

```
SH Event → EventBridge → Lambda/Connector → Ticketing System
                                     ↓
                               Analyst Updates Ticket
                                     ↓
                               Updates workflow in SH
```

This ensures that findings lifecycle is mirrored between SH and ticketing.

---

## 10 — Centralized Multi-Account Automation Using Organizations

With AWS Organizations + delegated admin, automation can be centralized.

### 10.1 — Admin account receives all org-wide events

No need to create Lambda/EventBridge rules in each member account.

### 10.2 — Admin executes remediation into member accounts

Using:

- Cross-account roles
- Step Functions with cross-account tasks

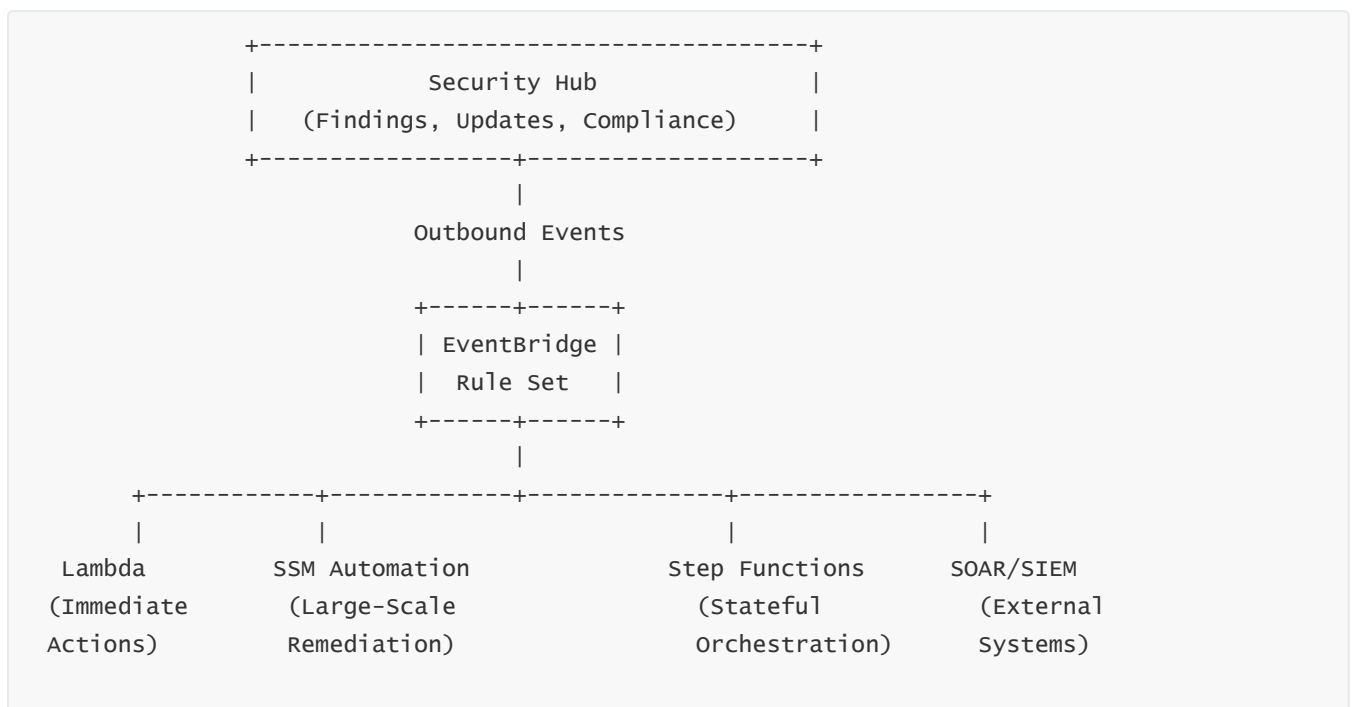
- SSM Automation with execution roles

## 10.3 — Benefits of Centralization

- Uniform remediation policies
- Consistent workflow tracking
- Cloud-wide security posture control
- Reduced operational overhead

Without centralized automation, multi-account remediation becomes unmanageable.

## 11 — Deep Internal Automation Pipeline Diagram



This shows how Security Hub becomes a **command-and-control layer** for security response.

## 12 — Automation for Compliance Controls & Standards

Compliance-driven automation is one of Security Hub's most powerful use cases.

### Example:

Control "S3.1 – Bucket allows public access" fails.

Automation flow:

```

Control Fails → SH emits event → EventBridge →
Lambda →
Block public access / set bucket policy / enable encryption →
SH workflow updated → Control re-evaluated → PASSED
  
```



This closes the loop between compliance detection and enforcement.

---

## **13 — Automation Challenges and How Security Hub Addresses Them**

---

### **13.1 — Large number of accounts**

Solution: Delegated admin + central automation.

### **13.2 — Large number of regions**

Solution: Region-specific event buses + optional multi-region orchestration.

### **13.3 — Consistency of fixes**

Solution: Step Functions + SSM templates.

### **13.4 — Avoiding infinite loops**

Security Hub ensures updated findings do not re-trigger rules unnecessarily through workflow state and dedup logic.

---

## **14 — Security Hub Automation Patterns**

---

### **14.1 — Reactive Pattern**

Triggered by a finding.

Useful for incident response.

### **14.2 — Preventative Pattern**

Triggered by compliance control failures.

Useful for governance enforcement.

### **14.3 — Proactive Pattern**

Triggered by predicted risks or external threat intelligence.

Useful for early warnings.

### **14.4 — Continuous Pattern**

Ongoing automation enforcing best practices across accounts.

---

## **15 — Mental Model Summary**

---

Security Hub automation is built on one simple idea:

“Every finding or compliance event is a structured, machine-readable signal that can automatically trigger the correct remediation workflow anywhere in the AWS environment.”

This is why Security Hub is not merely a reporting tool—it is the **automation control plane** that sits at the center of AWS security operations.

---

## 9 — Custom Insights, Queries, and Advanced Analytical Workflows in Security Hub

---

### 1 — What is an “Insight” in Security Hub and why does it exist?

---

An **Insight** in Security Hub is essentially a **saved, parameterized query** over your findings that is continuously evaluated and visualized.

- Conceptually, an Insight is *not* a static report; it is a **live analytical lens** that looks at your entire findings dataset (across services, accounts, and regions) and shows a summarized view based on filters and groupings you define.
- Security Hub’s internal architecture is built so that **findings are indexed** by attributes such as severity, product, account, resource type, standards, control IDs, status, tags, and many more. Insights sit on top of this index and continuously extract a particular **slice and aggregation** of the data.

So you can think of Insights as:

“Live dashboards built from query definitions over the ASFF findings store, continuously updated as new findings arrive or existing findings change.”

They are the core mechanism for analysts and architects to move from **raw findings noise** to **meaningful security narratives** like “Top 10 riskiest accounts”, “High-severity vulns in production”, “Controls failing in PCI workloads”, etc.

---

### 2 — Internal data model behind Insights: filter + group-by over indexed findings

---

Internally, Security Hub’s Insight engine works like a small **analytics layer** over the findings store. Every Insight has two conceptual components:

1. **Filter expression** – defines *which* findings are included.
2. **Aggregation (group-by) expression** – defines *how* those selected findings are grouped and counted.

– The **filter** is a structured condition: e.g., severity = HIGH, product = Inspector or GuardDuty, workflow != SUPPRESSED, resource type = `AwsEc2Instance`, account in `Prod OU`, etc.

– The **group-by** defines the dimension for summarization: e.g., group by `AwsAccountId`, or by `ResourceType`, or by `Severity.Label1`, or by a tag, and then count findings per group.

Conceptually:

```
Insight I:
  Filter:   F(finding) → true/false
  GroupBy:  G(finding) → key
  Metric:   Count or other simple aggregation over each key
```

Security Hub continuously maintains indexes so that these queries can be answered efficiently at scale.

---

## 3 — Indexing and query evaluation over the findings store

---

Security Hub's findings store is not just a flat JSON dump; it is optimized for **query and retrieval**.

Internally, for insights, Security Hub builds **indexes** on highly-used fields, for example:

- `AwsAccountId`
- `Severity.Label` and/or numeric severity
- `ProductArn`
- `Resource.Type` and `Resource.Id`
- `Compliance.Status`
- `RecordState` and `workflow.Status`
- `UpdatedAt`, `CreatedAt`
- `Types[]` (taxonomy types)

When an Insight runs, the query engine:

1. Uses these indexes to quickly identify candidate findings that match the filters.
2. Applies any more complex filters that cannot be fully supported by indexes.
3. Applies the group-by logic and metric aggregation (typically counts).
4. Returns a summarized list that the console renders as a chart/table.

This is why Insights can be updated **live** as findings change — they are not precomputed static reports.

---

## 4 — Types of built-in Insights vs custom Insights

---

Security Hub ships with **predefined (managed) Insights** that cover common security questions, for example:

- "Top 10 accounts with the most findings"
- "Failed findings by severity"
- "Recently active high-severity findings"
- "Findings associated with a specific standard or control family"

These are basically preconfigured filter + group-by definitions.

On top of that, you can build **Custom Insights**, where you:

- Define your own set of filters (what you care about).
- Choose your own grouping (how you want to see the data).
- Save it so that it becomes a persistent, live dashboard tile in the Security Hub console.

Custom Insights are crucial because **every organization's threat model and governance structure is unique**, and you want the console to reflect *your* way of looking at risk.

---

## 5 — The filter engine: how Security Hub selects the right subset of findings

---

The filter engine is essentially a **structured query language** for ASFF fields.

Internally, it looks like this:

```
Filter:
  Severity.Label in [HIGH, CRITICAL]
  AND ProductArn in [Inspector, GuardDuty]
  AND RecordState = ACTIVE
  AND Workflow.Status != SUPPRESSED
  AND AwsAccountId in [list-of-prod-accounts]
  AND Resource.Type = AwsEC2Instance
  AND UpdatedAt >= now() - 7 days
```

Each clause operates on **specific attributes** of the finding, often with:

- equality / inequality
- set membership (IN)
- prefix or substring matches on some string fields
- range filters on timestamp or numeric fields

The engine translates this structured filter into index lookups and efficient scanning operations inside the findings store.

Mentally, you can think:

“Security Hub has a mini query engine over findings; Insights are saved queries; the filter definition is the WHERE clause.”

---

## 6 — The grouping engine: how results are aggregated for visualization

---

Once the filter selects the relevant findings, Security Hub's Insight engine performs **aggregation**. The most common metric is **count of findings**, but the key piece is the **grouping dimension**.

Examples of grouping dimensions:

- Group by `AwsAccountId` → “Findings per account.”

- Group by `Region` → “Findings per region.”
- Group by `Resource.Type` → “Findings per resource category.”
- Group by `Severity.Label` → “Counts per severity level.”
- Group by `Compliance.Status` or `StandardsArn` → “Compliance posture.”

The engine essentially builds a map:

```
group_key → count
```

and then sorts (often by count descending) to show “top-N” groups. These are displayed as bar charts, tables, or other simple visualizations in the console.

## 7 — Insight lifecycle: creation, evaluation, refresh, and deletion

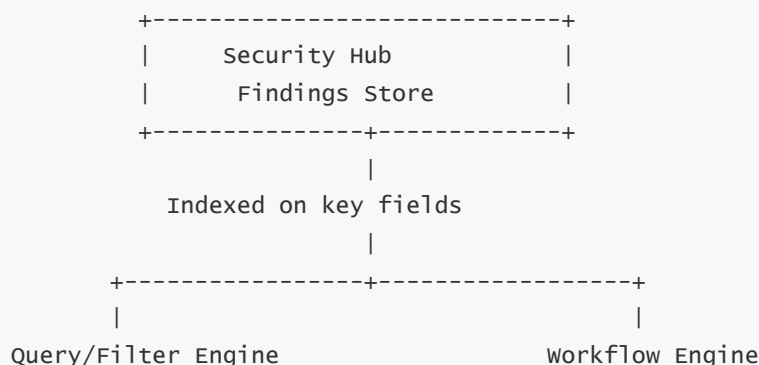
The lifecycle of a Custom Insight can be visualized as:

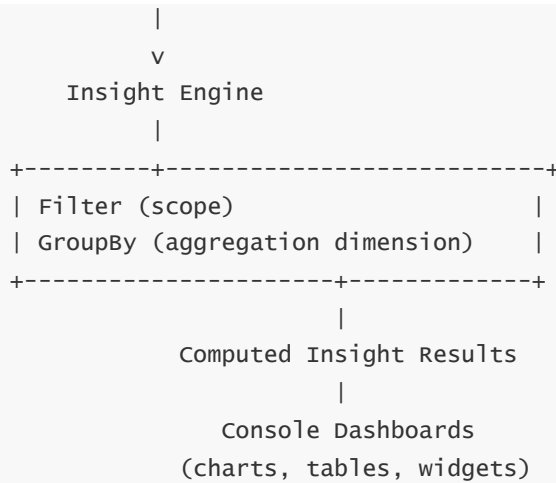
```
Define → Save → Continuous Evaluation → Adjust / Refine → (Optional) Delete
```

1. **Define** – You choose the filter and group-by.
2. **Save** – Security Hub stores this as a configuration object in the region/account.
3. **Continuous Evaluation** – As new findings come in or existing findings change, the Insight’s results update automatically.
4. **Adjust** – You refine the filter conditions or group-by fields as your needs evolve.
5. **Delete** – Removing the Insight does not affect findings; it only removes the view definition.

Internally, Security Hub stores these definitions in a **configuration store** separate from the findings store, and the query engine uses them on-demand.

## 8 — Full architecture diagram: findings → indexes → insights → dashboards





This logical structure is what lets Insights stay fast and responsive even at high finding volumes.

---

## 9 — Using Insights to design higher-level security questions

---

Where individual findings answer **“What is wrong with this single resource?”**, Insights help answer **“What is going wrong across my entire fleet?”**

Examples of higher-level questions you can model as Insights:

- “Which AWS accounts generate the most CRITICAL or HIGH severity findings in the last 7 days?”
- “What resource types are most frequently misconfigured across my org?”
- “Which controls in AWS Foundational Best Practices are failing most often?”
- “Where do I have long-lived, unresolved, high-severity compliance failures?”
- “Which partners (by ProductArn) are contributing the largest number of HIGH findings?”

In each case, you are simply combining:

- A scoped filter (time window, severity, product, status, account)
- A meaningful grouping dimension

The power comes from the **structured nature of ASFF**, which gives reliable fields for such queries.

---

## 10 — Insights in multi-account, organization-level deployments

---

In a **delegated admin** setup, Insights in the admin account operate over **aggregated findings from all member accounts in that region**.

- The Insight’s filter can constrain by `AwsAccountId`, `OrganizationalUnit-Tag` (if modeled via tags or mapping), `Region`, or `StandardsArn`.

- The group-by can be `AwsAccountId`, giving you **per-account views**, or `Resource.Type`, showing cross-account distributions.

So the same Insight engine which works in a single account also works as an **Org-wide analytics layer**, because all member findings are still just ASFF records in the admin's findings store.

---

## 11 — How Insights tie into automation and advanced workflows

---

Although Insights are primarily **visual analytic objects**, they influence automation and advanced workflows indirectly by:

- Helping you discover patterns that you then convert into **EventBridge rules** (e.g., you first observe that most high-risk problems are public S3 buckets; then you write rules to auto-remediate such findings).
- Helping you identify **key attributes** that should be used in automation conditions (e.g., group-by shows that a particular tag or OU is problematic).
- Serving as **monitoring dashboards** for automation performance (e.g., “High severity findings older than 24 hours should be zero; if count > 0, we know automation is failing”).

Thus, Insights guide the **design and tuning** of your remediation pipelines.

---

## 12 — Beyond Insights: exporting findings to external analytics systems

---

For **very advanced analytics** beyond what Security Hub Insights provide (e.g., complex time-series analysis, ML models, anomaly detection, long-term trend analysis), organizations often:

- Stream findings via EventBridge or other export mechanisms into:
- SIEM platforms
- Data lakes (S3 + Athena/Glue)
- Log analytics tools
- Security data platforms

Once in those systems, they can apply:

- Correlation with non-AWS data (on-prem logs, endpoint logs, identity logs).
- Long retention analytics.
- Custom dashboards and SOC playbooks.

Security Hub remains the **canonical source of normalized AWS security data**, and Insights remain the **first analytical layer**, while heavier data platforms do deep, multi-year analytics.

---

## 13 — Using Insights to support governance and executive reporting

---

From a governance perspective, Insights are used to create:

- “Top N” views per OU, per region, or per business unit.
- Monthly or weekly compliance summaries by standard.
- Heatmaps of resource types most frequently affected by issues.
- Progress tracking on remediation efforts (e.g., “High severity findings trend down 60% over the last quarter”).

Because Insights aggregate counts and states, they are ideal for **high-level, non-technical reporting** as well as for SOC team dashboards.

---

## 14 — Practical mental model for Insights in Security Hub

---

The simplest mental abstraction is:

“An Insight is a saved WHERE + GROUP BY query over the Security Hub findings table, evaluated continuously, visualized in the console, and covering all AWS and partner security signals in normalized form.”

- Findings are the **rows**.
- ASFF attributes are the **columns**.
- The filter is the **WHERE clause**.
- The group-by is the **GROUP BY clause**.
- The visualization is the **dashboard**.

Once we hold this mental model, building and reasoning about Insights becomes straightforward.

---

## 15 — How this all fits into the overall Security Hub analytics story

---

Putting everything together:

1. **Raw detections and compliance results** from AWS services and partners flow into Security Hub as ASFF findings.
2. The findings store **indexes and maintains** this data with states, timestamps, workflow, and compliance metadata.
3. The **Insight engine** sits on top as a **live, saved-query layer**, giving real-time summarized views that reflect the current posture.
4. These views guide humans (analysts, architects, CISOs) *and* guide the design of automation (EventBridge rules, Lambda, SSM, Step Functions, SOAR).
5. For even more advanced analytics, organizations export findings into external platforms, but Security Hub remains the **central source of normalized, structured, AWS security truth**.



That analytical layer—driven by Insights and query-like filters—is what elevates Security Hub from “a central inbox of security alerts” to “a decision and prioritization engine for cloud security.”

---

## 10 — Creating and Managing Custom Security Controls in AWS Security Hub

---

### 1 — Why Custom Controls Exist and What Problem They Solve

---

AWS Security Hub ships with hundreds of **managed** controls (CIS, PCI, FSBP, etc.), but every AWS environment has **unique governance requirements** that go beyond AWS’s default best practices.

Examples:

- “All Lambda functions must have environment variable encryption using KMS.”
- “All S3 buckets tagged with `Environment=Prod` must enforce versioning.”
- “Any IAM user older than 90 days with no MFA should be flagged.”
- “All ECR images must be scanned for vulnerabilities before deployment.”
- “Only approved KMS keys may be used for encryption across accounts.”

These are **organization-specific guardrails**, and AWS cannot pre-build controls for every custom business rule.

This is why Security Hub provides custom controls—allowing you to express **your own security checks**, aligned with your governance, threat model, and internal security policies.

A **custom control** is a **user-defined compliance rule** that:

1. Evaluates resources or account configuration.
2. Generates compliance results.
3. Produces ASFF findings.
4. Appears in compliance dashboards.
5. Aggregates across accounts.
6. Behaves like a managed control.
7. Integrates with automation pipelines.

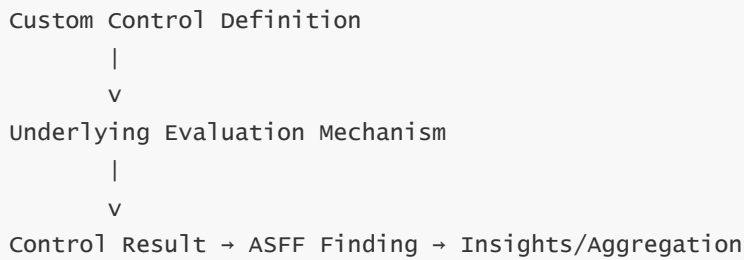
This effectively extends Security Hub into a **custom, organization-specific CSPM engine**.

---

### 2 — Internal Architecture of a Custom Control

---

A custom control is composed of three main layers:



Let's break these in detail.

## 2.1 — Control Definition Layer

This is a JSON/YAML-like object stored inside Security Hub that defines:

- **Name and description**
- **Severity**
- **Evaluation frequency**
- **Resource types to evaluate**
- **Enabled/disabled status**
- **Tags**
- **Standards group (optional custom standard)**
- **Underlying evaluation logic reference**

This is the “metadata wrapper” for the control.

## 2.2 — Evaluation Layer

The control is backed by one of the following evaluation engines:

### Option A — AWS Config Custom Rule

- Lambda-backed Config rule
- Custom logic written in Python/Node
- Config evaluates upon resource change or schedule
- Produces COMPLIANT/NON\_COMPLIANT

### Option B — Security Hub Native Custom Control (No Config)

- A new capability introduced to allow custom logic without Config
- Uses JSON-based evaluation logic
- Evaluates using internal Security Hub polling/API calls

### Option C — External Evaluator (Partner/On-Prem)

- You run evaluation logic in your own systems
- Push results to Security Hub using the BatchImportFindings API

Most enterprise environments use **Option A** for maximum control.

---

## 2.3 — Findings Generation Layer

Once the evaluation layer determines compliance per resource, Security Hub:

- Converts the result to a control compliance state ( `PASSED` , `FAILED` )
- Generates ASFF findings for failures
- Stores the findings in the regional findings store
- Sends events to EventBridge
- Surfaces results in dashboards and insights

Custom controls behave exactly like built-in controls at this layer.

---

## 3 — The Custom Control Authoring Workflow

---

Below is the internal workflow for creating a custom control:

### Step 1 — Define control metadata

Specify:

- Title
- Description
- Severity (LOW/MEDIUM/HIGH/CRITICAL)
- Resource type ( `AwsEC2Instance` , `AwsS3Bucket` , etc.)
- Evaluation frequency ( `DAILY` , `HOURLY` , `ON_RESOURCE_CHANGE` )
- Custom standard grouping (optional)

### Step 2 — Create evaluation logic

Choose one evaluation engine:

- Config rule (Lambda or GuardDuty-like logic)
- Security Hub JSON-rule-based evaluator
- External system evaluator

### Step 3 — Attach control to a custom standard (optional)

Allows grouping custom controls under an “Internal Best Practices” standard.

## Step 4 — Enable control across accounts and regions

Propagation path:

- Management account
- Delegated admin
- Member accounts

## Step 5 — Evaluation begins

Security Hub triggers evaluations based on the control's mechanism.

## Step 6 — Findings generated for failures

ASFF findings created for non-compliant resources.

## Step 7 — Dashboards and insights update

Compliance scores and insights reflect custom control results.

This makes custom controls a first-class component of your security governance framework.

---

# 4 — Config-Based Custom Controls (Deep Internal Mechanics)

---

Config custom rules are the backbone of powerful custom controls.

## 4.1 — Config Rule Structure

A Config rule consists of:

- Rule metadata
- Trigger type (change-triggered or periodic)
- Lambda function
- Input configuration snapshot
- Logic that returns COMPLIANT/NON\_COMPLIANT

## 4.2 — Example evaluation flow

```
Resource Change → Config detects new configuration →  
Triggers Lambda → Lambda logic evaluates →  
Returns NON_COMPLIANT for resource →  
Config publishes result → Security Hub ingests →  
Security Hub produces custom control finding
```

## 4.3 — Why Config custom rules are powerful

- Evaluate resource-level attributes
- API calls to inspect detailed configuration
- Logic can be arbitrarily complex
- Can embed organizational governance rules
- Can reference tags, naming conventions, architecture patterns

Every enterprise with strong governance uses Config custom rules to extend Security Hub.

---

## 5 — Security Hub Native Custom Controls (Rule-Based JSON Logic)

---

AWS recently introduced a more lightweight method for custom controls: **JSON-rule-based native controls**.

### 5.1 — How it works

Define:

- Resource types
- Required fields
- Conditions (e.g., “Encryption = enabled”)
- Severity
- Evaluation window

Security Hub evaluates using its internal data sources (API calls, Config data, metadata).

### 5.2 — Best suited for

- Simple checks (“field must equal X”)
- Tag validation
- Required configuration flags
- Resource enablement checks

### 5.3 — Not suited for

- Complex logic
- Multi-resource correlation
- Deep security heuristics

For deep governance, Config rules are still superior.

---

## 6 — External Evaluator Custom Controls (Advanced)

---

Organizations can evaluate controls externally and push results.

## 6.1 — How it works

External engine:

- Determines compliance
- Creates ASFF findings
- Uses `BatchImportFindings` to push results

## 6.2 — Common use cases

- On-prem to AWS governance controls
- Hybrid architecture checks
- Container/Kubernetes security checks
- Custom vulnerability or compliance engines
- Data center + AWS policy alignment

This method uses Security Hub as the **single pane of glass** for all controls, even outside AWS.

---

# 7 — Multi-Account and Multi-Region Propagation of Custom Controls

---

## 7.1 — Custom standard propagation

If you group custom controls inside a custom standard, enabling the standard in the delegated admin spreads:

- Control definitions
- Enablement
- Compliance evaluation

across all member accounts.

## 7.2 — Regional behavior

Controls must be enabled per region, just like AWS-managed standards.

## 7.3 — Benefits

- Organization-wide governance
  - Regionally isolated evaluation
  - Uniform posture enforcement
- 

# 8 — Custom Control Findings: ASFF Structure

---

A custom control finding is an ASFF record:

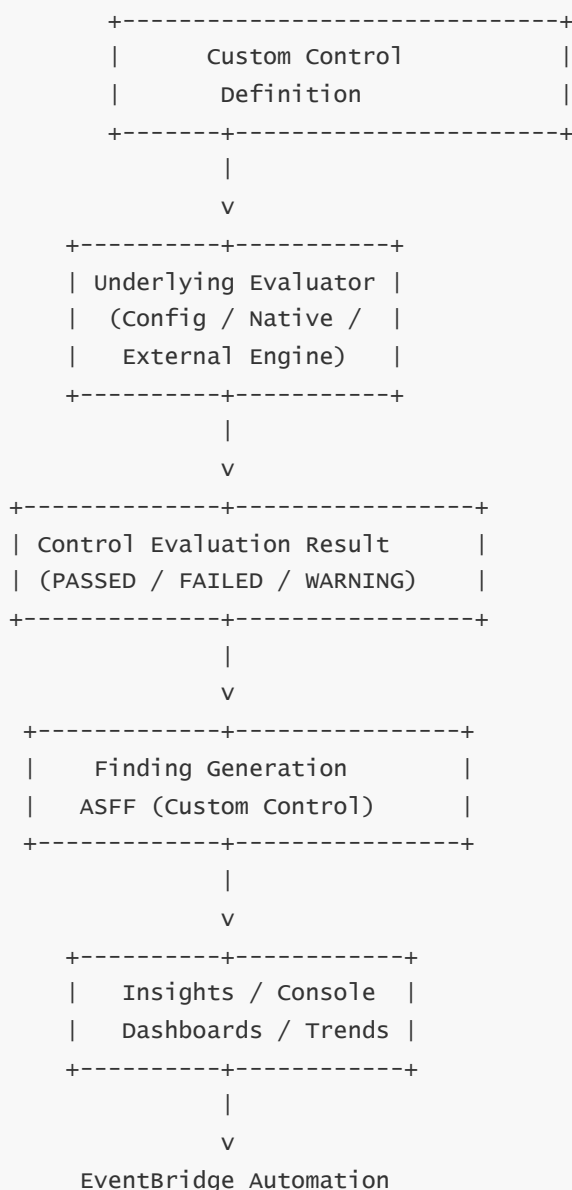
```

ProductArn: "arn:aws:securityhub:...:product/custom/<control-name>"
GeneratorId: "<CustomStandard>/<ControlId>"
Compliance: { Status: FAILED }
Severity: { Label: HIGH }
Resources: [ { Type: <AWSResourceType>, ... } ]
Workflow: NEW/NOTIFIED/RESOLVED/SUPPRESSED

```

Security Hub treats it **exactly** like a managed control finding.

## 9 — Full Internal Architecture Diagram for Custom Controls



This captures the complete custom control pipeline.

## 10 — Custom Control Lifecycle Management

---

### Create

Define metadata + evaluation logic.

### Enable

Deploy to member accounts and regions.

### Test

Verify evaluation correctness.

### Run

Allow continuous compliance checking.

### Tune

Refine logic, change severities, adjust resource scope.

### Retire

Disable control when governance requirement changes.

Security Hub maintains versioning for evaluation logic and control states as you update them.

---

## 11 — How Custom Controls Fit Into Organizational Governance

---

Custom controls allow organizations to express:

- Internal security policies
- Regulatory controls
- Architectural guardrails
- Platform design rules
- Workload-specific constraints
- Naming/tagging governance
- Compliance with industry frameworks that AWS does not provide

In a real environment, companies build dozens of custom controls like:

- “Prod accounts must not allow InternetGateways if VPC tag = Restricted.”
- “All RDS snapshots marked confidential must be encrypted with organization KMS keys.”
- “IAM roles must not be assumable by other AWS accounts except via whitelist.”

Security Hub becomes the **platform for policy-as-code** across the AWS footprint.

---



## 12 — Mental Model Summary

---

To simplify everything:

“A custom control is a governance rule defined by you, evaluated by Config/Security Hub/third-party logic, producing ASFF findings like AWS-managed controls, aggregating across accounts, and driving automation.”

This framework turns Security Hub into a **customizable compliance engine** suitable for any enterprise's unique security requirements.

---

## 11 — Deep Dive into ASFF Severity, Scoring Models, and Risk Interpretation

---

### 1 — Why Severity and Scoring Matter in Security Hub's Architecture

---

In any large-scale cloud environment, thousands — sometimes tens of thousands — of findings flow into Security Hub from:

- GuardDuty (threat detections)
- Inspector (vulnerabilities)
- IAM Access Analyzer (identity exposure)
- AWS Config (compliance drift)
- Security Hub's native controls
- Partner tools (CSPM, CWPP, EDR, DLP, TI platforms)

Each source has its own way of expressing severity.

Without normalization, your security team would drown in incompatible scoring systems.

Security Hub solves this by creating a **unified risk model** using the **AWS Security Finding Format (ASFF)**, converting all vendor-specific severities into a consistent structure.

This allows:

- Uniform analysis
- Comparable risk views
- Prioritization across heterogeneous tools
- Cross-account risk dashboards
- Automated workflows based on unified severity thresholds
- Proper SLA mapping across the organization

Severity normalization is the **core mechanism** that allows Security Hub to act as a central prioritization engine.

---

## 2 — ASFF Severity Object: The Heart of the Risk Model

The ASFF severity block looks like this conceptually:

```
Severity
├─ Label          (INFORMATIONAL | LOW | MEDIUM | HIGH | CRITICAL)
├─ Normalized     (0 - 100 score)
└─ Original       (vendor-provided numeric or qualitative value)
```

Each field serves a purpose.

### 2.1 — Severity.Label

This is the primary field used for:

- Dashboards
- EventBridge automations
- Insights grouping
- Cross-account analytics
- Filtering
- Ticket prioritization

Standardized labels:

- **INFORMATIONAL**
- **LOW**
- **MEDIUM**
- **HIGH**
- **CRITICAL**

### 2.2 — Severity.Normalized

A numeric field on a **0–100 scale** representing AWS's normalized severity.

Typical mapping:

- 0–19: LOW
- 20–39: MEDIUM
- 40–69: HIGH
- 70–100: CRITICAL

This allows cross-tool comparisons even if vendor scoring systems differ.

## 2.3 — Severity.Original

Represents the vendor's native severity.

Examples:

- Inspector CVSS score = 9.8
- GuardDuty severity = 0.3
- Third-party tool severity = "EXTREME RISK"
- Partner DLP platform = "Level 5"

Security Hub keeps the raw value for traceability.

---

## 3 — How Severity Normalization Works Internally

Security Hub normalizes severity in four stages:

### 3.1 — Stage 1: Vendor Severity Intake

Partner or AWS native services send a raw severity value:

Examples:

- Inspector: CVSS 7.2
- GuardDuty: severity = 5.0
- Custom partner: "High Risk"
- Config: "NON\_COMPLIANT" (no score)

Security Hub captures these into `Severity.Original`.

---

### 3.2 — Stage 2: Severity Mapping Rules Applied

Security Hub maintains mapping logic for each source:

Example mapping for GuardDuty:

- 0.0–3.9 → LOW
- 4.0–6.9 → MEDIUM
- 7.0–8.9 → HIGH
- 9.0+ → CRITICAL

Example mapping for Inspector (CVSS):

- CVSS 0.1–3.9 → LOW
- 4.0–6.9 → MEDIUM
- 7.0–8.9 → HIGH
- 9.0–10 → CRITICAL

Example mapping for Config:

- FAILED → MEDIUM (default)
- FAILED + sensitive resource → HIGH

Partner tools also provide their mapping; Security Hub applies AWS's translation rules.

---

### 3.3 — Stage 3: Severity.Label Assigned

After mapping, Security Hub assigns:

- INFORMATIONAL
  - LOW
  - MEDIUM
  - HIGH
  - CRITICAL
- 

### 3.4 — Stage 4: Severity.Normalized Computed

AWS assigns a normalized numeric value (0–100) which acts as a **cross-service severity score**.

|          |   |        |
|----------|---|--------|
| CRITICAL | → | 70–100 |
| HIGH     | → | 40–69  |
| MEDIUM   | → | 20–39  |
| LOW      | → | 1–19   |
| INFO     | → | 0      |

This numeric value allows deterministic ordering in analytics.

---

## 4 — Why AWS Needs Two Parallel Severity Systems

Having *both* qualitative labels and numeric scores solves key problems:

### Qualitative labels

- Used for automation thresholds
- Human friendly
- Compliance reporting
- SOC playbooks
- Dashboard grouping

## Numeric normalized score

- Precise ranking
- Sorting
- Risk trending
- Fine-grained insights
- Machine-driven prioritization

This duality allows both human and machine workflows to operate effectively.

---

## 5 — Resource-Type Weighting: Context-Aware Severity Interpretation

---

Not all findings with the same severity represent equal risk.

Security Hub uses **resource context** to adjust risk interpretation even if the severity label is the same.

Examples:

- A HIGH finding on a **public S3 bucket** is more dangerous than a HIGH finding on an isolated EC2 instance.
- A CRITICAL CVE on a production autoscaling group resource is more urgent than the same CVE in a dev environment.

Security Hub does not automatically modify severity based on resource type, but its Insights and workflow logic allow you to **overlay severity with asset criticality**, e.g.:

```
Severity = HIGH
Resource.Type = AwsS3Bucket
Environment = Prod
→ escalate urgency to maximum
```

This contextual overlay is implemented at the observation/automation layer.

---

## 6 — Confidence and Criticality: Complementary Risk Dimensions

---

The ASFF model includes two additional fields:

```
Confidence → Likelihood the finding is accurate
Criticality → Importance of the affected resource
```

These fields allow enhanced triage.

## 6.1 — Confidence

Used heavily in threat-driven tools (GuardDuty, EDR vendors).

Meaning:

- High confidence → low chance of false positive
- Low confidence → anomaly or heuristic-based signal

Automation systems often filter out lower-confidence findings.

## 6.2 — Criticality

Represents how important the resource is.

Examples:

- A critical payment system EC2 instance
- A production RDS database with sensitive customer data
- A mission-critical IAM role

You can tag resources with business criticality and feed into automation.

---

# 7 — Severity Interpretation Across Service Types

---

Different AWS native tools produce findings with different inherent methods.

## 7.1 — GuardDuty Severity Interpretation

GuardDuty severity reflects threat impact and confidence:

- High: malicious behavior with strong evidence
- Medium: suspicious behavior
- Low: benign anomalies or broad scans

## 7.2 — Inspector Severity Interpretation

Inspector's severity is rooted in CVSS scoring:

- HIGH = critical exploitability
- MEDIUM = moderate exploitability
- LOW = minor or informational vulnerabilities

Security Hub normalizes these scores for cross-service comparison.

## 7.3 — AWS Config Severity Interpretation

Config findings represent **misconfigurations**, not threats.

Security Hub maps failure severity based on:

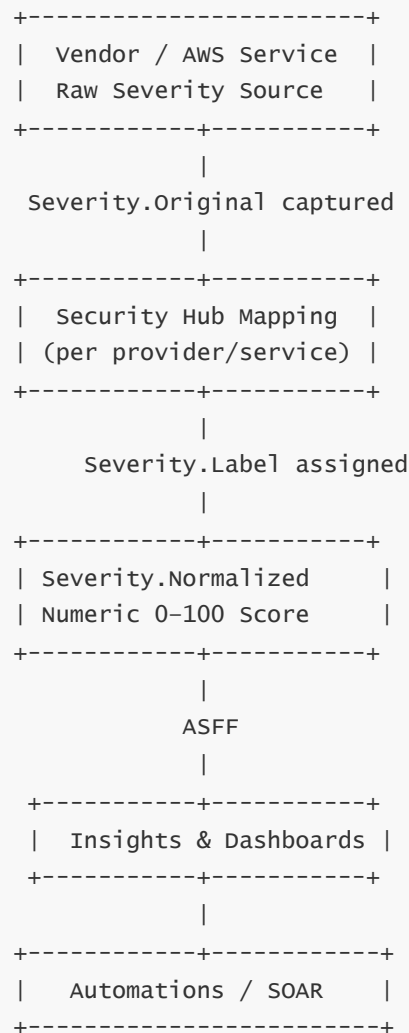
- Compliance impact

- Security criticality
- Resource type

---

## 8 — Multi-Layer Severity Diagram: End-to-End Normalization Flow

---



This is how Security Hub unifies the risk landscape.

---

## 9 — Combining Severity, Confidence, Criticality, and Compliance

---

In mature orgs, risk scoring is not simply “severity of the finding,” but:

```
RiskScore = f(severity, confidence, criticality, compliance impact)
```

## Example:

- Severity = MEDIUM
- Confidence = HIGH
- Criticality = HIGH
- Compliance impact = Hits PCI requirement

This may be operationally treated as CRITICAL even if the severity label is MEDIUM.

Security Hub facilitates this by exposing all dimensions in ASFF.

---

## 10 — Severity in Cross-Account Aggregation

---

Aggregated dashboards allow you to:

- See which accounts generate the most CRITICAL/HIGH findings
- Identify “noisy accounts”
- Prioritize remediation where severity clusters
- Detect outlier behavior
- Identify high-risk regions

This uses the normalized severity values, ensuring fairness across accounts.

---

## 11 — Automations Driven by Severity

---

Severity is the primary driver for EventBridge-based remediation.

Examples of automation triggers:

- Any CRITICAL finding → Notify SOC immediately
- Any HIGH compliance failure → Trigger remediation Lambda
- Any MEDIUM vulnerability older than 7 days → Create Jira ticket
- LOW severity findings → Only log for historical tracking

Severity acts as the *switch* for automated playbook logic.

---

## 12 — Severity Trends and Time-Based Analytics

---

Security Hub tracks:

- Aging high-severity findings
- High-severity trends
- Mean time to resolution per severity
- Region-level severity distributions
- OU-level severity patterns



These analytics help security leadership measure posture improvement.

---

## 13 — Severity in Custom Controls

---

Custom controls allow you to explicitly choose severity.

Examples:

- Tagging violations (MEDIUM)
- Encryption disabled (HIGH)
- Unrestricted IAM trust (CRITICAL)
- Missing logging (HIGH)

This integrates custom governance into the main risk hierarchy.

---

## 14 — Partner Tool Severity Normalization

---

Security Hub absorbs a variety of vendor scoring systems:

- Tenable (Tenable score, CVSS)
- Qualys severity
- Lacework severity
- Wiz severity
- Prisma Cloud severity
- Datadog security findings
- Rapid7 severity

Security Hub normalizes all of them so your dashboards remain coherent.

---

## 15 — Mental Model Summary

---

You can summarize the entire severity model in one sentence:

“ASFF converts every vendor’s proprietary severity system into a unified AWS severity label and numeric score, enabling consistent analytics, prioritization, cross-account comparison, and automated remediation.”

This consistency is the backbone of Security Hub’s risk interpretation system.

---

## 12 — Security Hub Cost Model, Optimization Strategies, and Operational Efficiency Architecture

---

---

# 1 — Why Cost Modeling Matters for Security Hub

---

Security Hub is often deployed across:

- dozens
- hundreds
- or even thousands of AWS accounts,  
with multi-region enablement.

This creates a **massive multiplier effect** on cost.

Security Hub is architected to give **organization-wide visibility**, but the cost must be understood and controlled to avoid surprises.

Because Security Hub integrates with Config, GuardDuty, Inspector, Macie, Firewall Manager, and partner tools, its cost model intersects with the cost models of many other services.

Understanding cost at a deep architectural level requires analyzing:

- how Security Hub counts findings
- how Config evaluations impact cost indirectly
- how multi-region expansion affects spend
- how partner integrations introduce chargeable units
- how workflow churn increases mutation traffic
- how control evaluations scale with resource count
- how automation pipelines indirectly consume Lambda/SSM/Step Functions resources

Security Hub is *not* one of the most expensive security services, but misconfiguration or unnecessary multi-region activation can multiply cost dramatically.

---

## 2 — The Core Security Hub Cost Components

---

Security Hub's billing is composed of two main charge models:

### 2.1 — Cost Component 1: Ingestion of Findings

Security Hub charges you for:

- number of **finding events ingested**
- number of **finding update events**

Every ASFF record from:

- GuardDuty
- Inspector
- IAM Access Analyzer
- AWS Config
- Native controls

- Partner tools

...counts as ingestion.

Note:

**Updates** also count as billable events, but at a lower rate than new imports.

---

## 2.2 — Cost Component 2: Security Hub Controls / Compliance Checks

You pay for the number of *compliance checks* Security Hub performs.

A “compliance check” is:

one control evaluating one resource once.

If a control applies to 1,000 S3 buckets and triggers daily, that is:

1,000 evaluations per day  
30,000 evaluations per month

Multiply that by:

- the number of controls in the standard,
- the number of regions enabled,
- the number of accounts,
- and the total resource count,

and you begin to see how cost scales exponentially.

---

## 3 — Multi-Layer Architectural Cost Model Breakdown

Let us visualize the conceptual cost stack:

```
+-----+
| Security Hub Resource Evaluations | (Controls)
+-----+
| Security Hub Findings Ingestion   |
+-----+
| Underlying Service Costs          |
| (Config, GuardDuty, Inspector, etc.) |
+-----+
| Automation Costs                  |
| (Lambda, SSM, Step Functions)     |
+-----+
| Partner Tool Costs (if ingesting) |
+-----+
| Cross-Account Aggregation Overhead |
```

+-----+

Each layer contributes differently to cost.

---

## 4 — Cost Amplification in Multi-Account + Multi-Region Deployments

---

Security Hub is **regional**, not global.

This means:

If you enable Security Hub in **15 regions** and have **200 accounts**, then:

- You have  $15 \times 200 = 3,000$  independent Security Hub deployments
- Compliance controls run in each region
- Findings ingestion occurs in each region
- Admin account aggregates per region

This can multiply cost dramatically.

Organizations often **reduce regions** to control cost.

---

## 5 — Deep Breakdown: Cost of Findings Ingestion

---

### 5.1 — What counts as a “finding ingestion”?

Every time a finding enters Security Hub through ASFF import:

```
New ASFF finding = 1 ingestion
Updated finding = 1 update ingestion
```

### 5.2 — High-volume sources

Services that generate high-volume findings:

- Inspector (especially container scanning)
- GuardDuty (unusual API pattern + anomaly detections)
- Macie (data classification)
- Partner CSPM/CWPP tools

Inspector container scanning can generate **thousands of vulnerability findings per image**, and each update counts.

## 5.3 — Partner tools can drastically increase ingestion cost

If a CSPM tool pushes 1 finding per resource violation per day across 100 accounts:

- 150,000 findings/month → very high cost impact

Thus, partner integrations need careful cost governance.

---

## 6 — Deep Breakdown: Cost of Control Checks

Security Hub's control evaluation cost is often larger than ingestion cost.

### 6.1 — Evaluation formula

For each control:

```
Total evaluations = Number of resources in scope × evaluation frequency
```

### 6.2 — Example

- Control: "S3 buckets must block public access"
- 2,000 S3 buckets
- Evaluated daily

Then:

```
60,000 evaluations per month *for one control*
```

A standard (like AWS FSBP) may include 140 controls.

If many apply to S3 or EC2, cost can explode.

---

## 7 — Indirect Cost Contributors

These are non-Security Hub resources that indirectly determine cost.

### 7.1 — AWS Config costs

Security Hub heavily relies on Config managed rules.

Config rules generate:

- Configuration change recordings
- Evaluation cycles
- Snapshot evaluations (chargeable)

Config is often **the largest hidden cost** behind Security Hub.

## 7.2 — Inspector costs

Inspector scans generate large numbers of findings.

More findings → more Security Hub ingestion cost.

## 7.3 — Automation costs

EventBridge → Lambda → SSM → Step Functions

If Security Hub produces many findings, your automation pipeline may explode in cost.

## 7.4 — Partner tools

Some push massive numbers of findings.

## 7.5 — Cross-region duplication

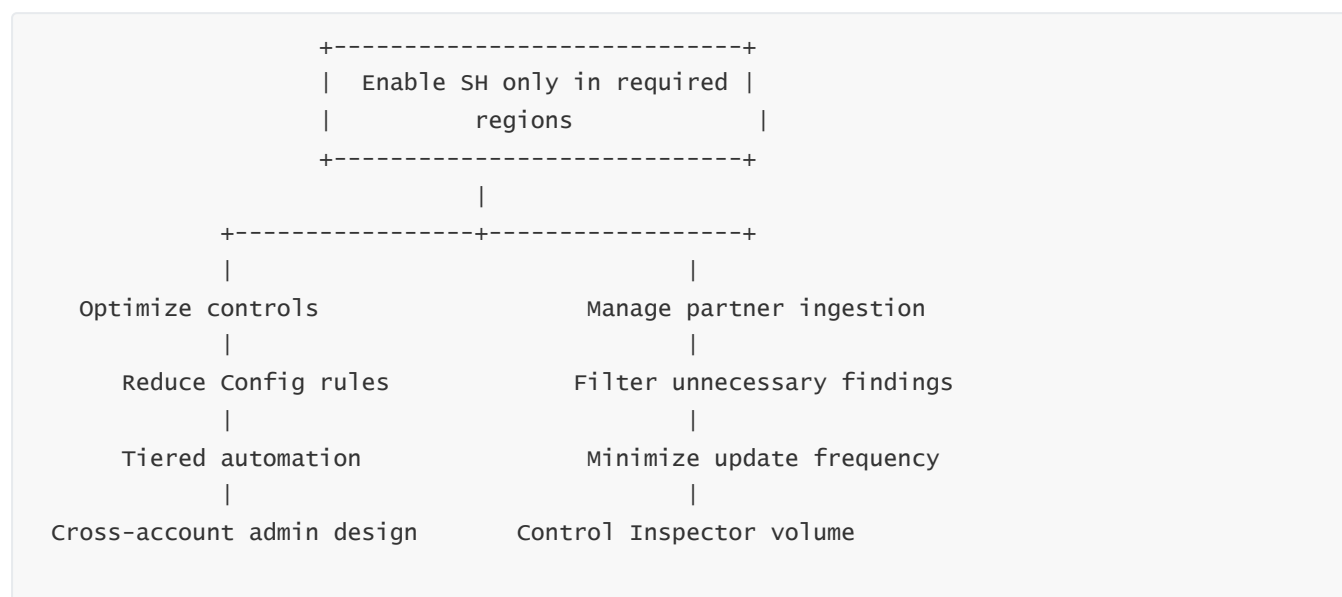
Running Security Hub in unnecessary regions multiplies costs.

---

# 8 — Cost-Optimized Architecture: The Practical Model

---

Let's construct an optimized architecture:



This architecture aims to reduce unnecessary evaluations and ingestion.

---

# 9 — Optimization Strategy 1: Control Scope Tuning

---

## 9.1 — Disable controls irrelevant to your environment

For example:

If you do not use CloudFront, disable CloudFront controls.

## 9.2 — Reduce control frequency

Some native controls allow tuning of evaluation frequency.

Less frequent = fewer evaluations = cost savings.

## 9.3 — Target controls by OU or account

Use delegated admin to enforce controls only where needed.

---

# 10 — Optimization Strategy 2: Region Selection

---

### Most enterprises require only 3–5 regions:

- ap-south-1 (India)
- us-east-1
- us-west-2
- eu-west-1

If you enable Security Hub in **20 regions**, your cost increases 20×.

Reduce region scope aggressively.

---

# 11 — Optimization Strategy 3: Config Rule Minimization

---

### Config drives the majority of control evaluation cost.

Steps:

1. Disable Config in unused regions.
2. Reduce number of active Config rules.
3. Move heavy Config custom rules to event-driven Lambda functions.
4. Aggregate Config events using custom logic instead of rules.

Config tuning can reduce Security Hub costs **by 50–80%**.

---

# 12 — Optimization Strategy 4: Inspector Volume Rationalization

---

Inspector can generate thousands of findings per resource.

### How to control:

- Limit container scanning frequency
- Disable scanning on dev accounts
- Restrict scanning to critical registries

- Use build-time scanning instead of runtime scanning
- Filter out low CVSS findings from ingestion

Inspector tuning dramatically reduces ingestion costs.

## 13 — Optimization Strategy 5: Partner Tool Noise Reduction

Partner tools often flood Security Hub with noisy or duplicate findings.

### Solutions:

- Configure partner tool to send only HIGH/CRITICAL
- Disable daily re-ingestion of unchanged findings
- Filter findings by tags or environments
- Use selective integration
- Send only deltas, not full scans

Most partner tools allow such filtering, reducing ingestion cost by 70–95%.

## 14 — Architectural Diagram: Cost Flows + Optimizations



This diagram shows where the levers of cost control lie.



## 15 — Practical Cost Governance Model for Security Hub

---

A functional enterprise governance model includes:

1. **Cost baselining**

Track ingestion and control evaluations per account/region.

2. **Optimization guardrails**

Force accounts to use only approved SH regions.

3. **Partner integration governance**

Partner tools cannot push unlimited data.

4. **Pipeline suppression**

Automatically archive or suppress low-value findings.

5. **Tiered implementation**

Use Security Hub only in critical workloads; keep dev/test light.

6. **Monthly review dashboards**

“Top accounts by Security Hub cost,” etc.

7. **Cross-team ownership structure**

Ensure Config, Inspector, SH, and automation teams collaborate.

---

## 16 — Mental Model Summary

---

You can summarize the entire cost architecture as follows:

“Security Hub’s cost is driven by findings ingested and compliance evaluations executed.

These scale with region count, account count, resource count, Config rules, Inspector volume, partner tool noise, and automation churn.

Cost optimization is achieved by reducing unnecessary regions, controls, Config rules, partner findings, and heavy automation.”

This is the cost-centric lens required for operating Security Hub at cloud scale.

---

## 13 — Security Hub Workflow Management, Finding Lifecycle, and Operational State Transitions

---

### 1 — Why Workflow Management Exists in Security Hub

---

Security Hub ingests findings from dozens of AWS services and partner platforms. These findings are not static objects; they represent **ongoing security issues** that must be triaged, investigated, assigned, remediated, validated, and ultimately resolved.

In a real SOC environment, thousands of findings may accumulate. Without workflow management, Security Hub would simply be a **data lake of alerts** with no operational tracking or process enforcement.

Workflow management turns Security Hub from a **security inbox** into a **case management and investigation system**, enabling:

- Consistent tracking of investigation progress
- Assignment of ownership
- Integration with SOAR / ITSM / ticketing platforms
- Automated lifecycle updates
- Suppression of noisy findings
- Multi-account workflow sync

This system allows findings to move through well-defined **states**, enabling security teams to maintain clarity over what is new, what is being worked on, what has been resolved, and what should be ignored.

---

## 2 — The Complete Finding Lifecycle in Security Hub

---

Security Hub findings move through a **strict, closed-loop lifecycle**, controlled via the `workflow.status` field in ASFF.

There are four main states:

```
NEW → NOTIFIED → RESOLVED → SUPPRESSED
```

Security Hub enforces this sequence to maintain data integrity and operational clarity.

Let us examine each in depth.

---

### 2.1 — NEW

This is the default state for all newly ingested findings.

Meaning:

- No one has responded yet
- No workflow actions applied
- Not assigned, not triaged

`workflow.status = NEW` is what drives:

- SOC dashboards
- High-priority queues
- First-response automations
- EventBridge “fresh alert” triggers

In this state, the finding is considered **active, unacknowledged**, and **requiring attention**.

## 2.2 — NOTIFIED

This state indicates that the SOC or automation pipeline has **acknowledged** the finding.

Typical transitions to NOTIFIED result from:

- Lambda automation
- SOAR enrichment step
- Ticket creation in Jira/ServiceNow
- Analyst manually marking it
- Event-driven workflows

Meaning:

- “Someone has been alerted; work is in progress.”

In large environments, NOTIFIED is essential for suppressing repeated re-alerts.

---

## 2.3 — RESOLVED

RESOLVED indicates that a fix has been implemented, and remediation activity is concluded.

This state is usually reached when:

- Automation applies a fix
- Analyst confirms manual remediation
- Resource is deleted
- Risk accepted and mark-as-resolved enforced

Important:

**RESOLVED does not remove the finding.**

The finding remains, but marked as “addressed.”

If the underlying resource still violates a control, Security Hub may reopen a NEW finding on the next evaluation cycle.

This ensures that compliance drift is not ignored.

---

## 2.4 — SUPPRESSED

SUPPRESSED is the state that indicates:

- Finding is irrelevant,
- Accepted risk,
- False positive,
- Will not be addressed,
- Not applicable to workload.

Security Hub removes SUPPRESSED findings from:

- Insights
- Dashboards
- Severity statistics
- Automation signals

Suppression must be used sparingly—excessive suppression hides critical signals.

### 3 — Internal Workflow State Machine: Security Hub Enforcement Logic

Security Hub enforces a strict state machine behind the scenes.



Rules:

- NEW → NOTIFIED (allowed)
- NOTIFIED → RESOLVED (allowed)
- NOTIFIED → SUPPRESSED (allowed)
- RESOLVED → NEW (allowed when issue reappears)
- SUPPRESSED → NEW (allowed when new finding occurs)

Security Hub **never deletes** findings—resolution or suppression only changes state, not existence.

### 4 — Workflow Change Propagation and Synchronization

Workflow changes are **not local-only**.

Security Hub synchronizes workflow states:

- across member accounts

- with delegated admin
- with partner tools (if integrated)
- with ticketing systems
- with automation pipelines
- with event-driven updates

This enables:

- Cross-account workflow consistency
- SOC-level global triage
- Consistent automation behavior
- Avoiding duplicated investigation work

Example:

If the admin account marks a finding as RESOLVED, the member account's Security Hub view also updates.

---

## 5 — ASFF Workflow Object: The Technical Foundation

In ASFF, workflow state is represented as:

```
"workflow": {  
  "Status": "NEW" | "NOTIFIED" | "RESOLVED" | "SUPPRESSED"  
}
```

This field is persisted in the findings store and is used:

- by EventBridge filters
- by Insights
- by partner integrations
- by automations
- by cross-account aggregation logic

---

## 6 — Automation-Driven Workflow Management

Workflow automation is common in large enterprises.

### Example 1 — NEW → NOTIFIED

Triggered by EventBridge rule:

- Raised when new critical finding arrives
- Lambda updates finding's workflow to NOTIFIED
- SOAR tool receives the updated finding

## Example 2 — NOTIFIED → RESOLVED

Triggered when remediation Lambda finishes executing.

Security Hub update:

```
UpdateFindings  
workflow.Status = RESOLVED
```

## Example 3 — NOTIFIED → SUPPRESSED

Triggered by:

- false positive logic
- tag-based suppression rules
- resource exceptions
- architect-approved risk acceptance

## Example 4 — RESOLVED → NEW

Auto-triggered when issue reappears.

---

## 7 — Role of Workflow in Ticketing Systems (Jira, ServiceNow)

Workflow sync allows:

- NEW → creates ticket
- NOTIFIED → updates ticket with acknowledgment
- RESOLVED → closes ticket
- SUPPRESSED → marks ticket as WONTFIX

This provides end-to-end lifecycle integrity across systems.

---

## 8 — Workflow and Rate-Limited Re-Alert Mechanisms

Without workflow states, repeated updates to a finding would create alert floods.

Security Hub uses workflow state to prevent:

- Duplicate notifications
- Duplicate SOAR triggers
- Automation loops
- Alert fatigue

Example:

An Inspector finding may update 20 times during scanning.

Security Hub uses workflow state rules to avoid firing EventBridge triggers every time.

---

## 9 — Incident Response Integration

---

Workflow states act as signals in IR pipelines:

- NEW = trigger investigation
- NOTIFIED = investigation initiated
- RESOLVED = investigation closed
- SUPPRESSED = no investigation necessary

SOAR tools often implement IR playbooks mapped to these states.

---

## 10 — Building Workflow Policies (Organization Level)

---

Enterprises define workflow governance policies:

### For critical workloads:

- NEW must be moved to NOTIFIED within 15 minutes
- RESOLVED SLA is 4 hours
- SUPPRESSED must require approval

### For non-prod workloads:

- NOTIFIED happens via automation
- RESOLVED only when automated remediation succeeds
- SUPPRESSED allowed for dev-only exceptions

Security Hub enforces these policies via automation.

---

## 11 — Workflow for Compliance vs Threat Findings

---

Workflow behavior differs depending on finding category.

---

### 11.1 — Compliance Findings

Generated from controls (Config, SH native).

Behavior:

- RESOLVED means misconfiguration fixed
- Issue may reappear
- SUPPRESSED represents accepted exception

Compliance drift occurs frequently, making workflow states vital.

## 11.2 — Threat Findings

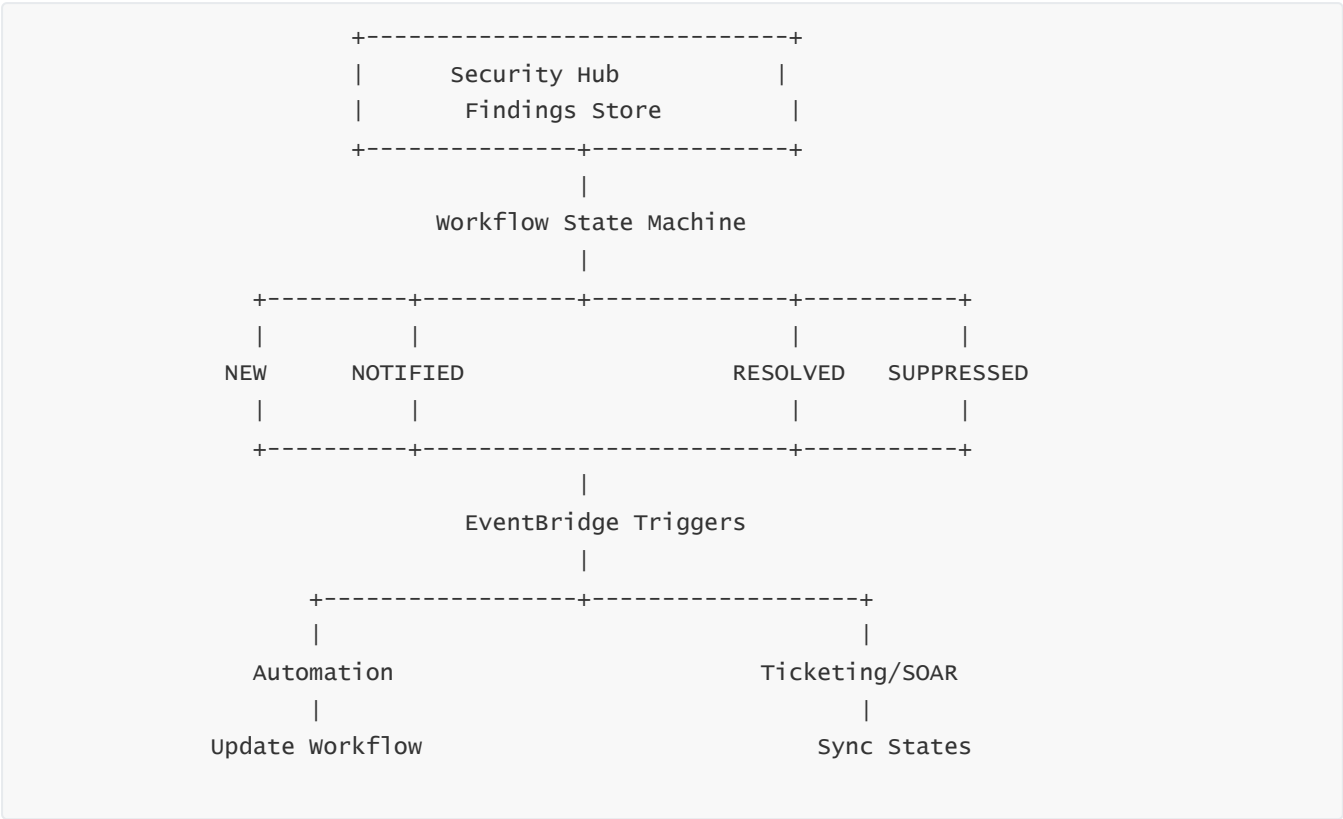
Generated by GuardDuty, EDR, CWPP, etc.

Behavior:

- NEW triggers immediate response
- NOTIFIED indicates SOC has taken ownership
- RESOLVED often requires multi-step validation
- SUPPRESSED rarely used except for false positives

Threat workflows are urgency-driven.

## 12 — Multi-Layer Diagram: Full Finding Workflow System



This depicts the lifecycle, automations, and integration points.

## 13 — How Workflow Enables Large-Scale Cloud Governance

With thousands of accounts and millions of findings, workflow states enable:

- **Prioritization** of active issues
- **Tracking** of investigation status
- **Auditing** of remediation



- **Historical analysis** of security posture
- **Lifecycle visibility** across regions
- **RCA documentation**
- **Compliance auditor evidence**

Workflow is the *governance glue* that transforms raw detection into structured ongoing operations.

---

## 14 — Failure Scenarios and How Workflow Handles Them

---

### Scenario 1 — Resource deleted before remediation

Security Hub can still mark finding RESOLVED.

### Scenario 2 — Automation fails

Workflow remains NOTIFIED; operations are alerted.

### Scenario 3 — Misconfiguration fixed but reintroduced

Security Hub reopens finding as NEW.

### Scenario 4 — Threat reappears

GuardDuty produces new finding → reset to NEW.

### Scenario 5 — Analyst mistakenly suppresses a critical finding

EventBridge monitoring can override suppression if conditions indicate risk.

---

## 15 — Mental Model Summary

---

You can think of the workflow system as:

“A built-in, distributed ticketing and state-tracking engine that ensures every finding from every AWS account moves through a clear lifecycle of validation, investigation, remediation, and closure—across all regions and all integrated security tools.”

Workflow states provide the foundational structure for **operationalizing** Security Hub at enterprise scale.

---

## 14 — Aggregation Insights: Cross-Region, Cross-Account Analysis and Enterprise Security Posture Metrics

---

# 1 — Why Aggregation Insights Exist: Turning Millions of Findings into Strategic Intelligence

---

In a modern AWS enterprise, it is common to operate:

- 50+ AWS accounts
- 10+ AWS regions
- 100,000+ AWS resources
- 20+ security services and partner tools feeding findings
- Millions of historical and current findings over time

Raw findings alone provide **no meaningful context**.

Security Hub's Aggregation Insights layer transforms "mass alert data" into **structured, executive-level risk visibility**.

Aggregation Insights exist to answer the critical questions that security leaders ask:

- Which accounts are the riskiest?
- Which OUs repeatedly violate governance requirements?
- Which regions have the worst security hygiene?
- What categories of issues dominate our environment?
- Are vulnerabilities trending up or down?
- Are we improving over time?
- Which workloads consistently drift?
- Which applications or tags represent concentrated risk?

Aggregation Insights turn Security Hub into an **organization-wide security analytics platform**, not just a local alert viewer.

---

## 2 — The Internal Architecture Behind Aggregation Insights

Aggregation in Security Hub is built on three core layers:

1. Regional Findings Stores (per account/region)
2. Regional Aggregators (delegated admin per region)
3. Insight Engine (aggregation + analytics)

Let's break these down.

---

## 2.1 — Layer 1: Regional Findings Stores

Each account and each region has its **own** Security Hub findings store.

This ensures:

- Data sovereignty
- Performance isolation
- Blast-radius containment
- Region-level failure independence

All findings remain stored in their originating region.

---

## 2.2 — Layer 2: Regional Aggregators

A delegated admin account becomes the **aggregation point** for each region:

```
Member Account A (Region X)
Member Account B (Region X)
Member Account C (Region X)
      ↓
Delegated Admin (Region X)
```

This admin account receives all findings and compliance states for that region.

Aggregation happens *per region*; there is **no built-in cross-region replication**.

---

## 2.3 — Layer 3: Insight Engine (Org-level)

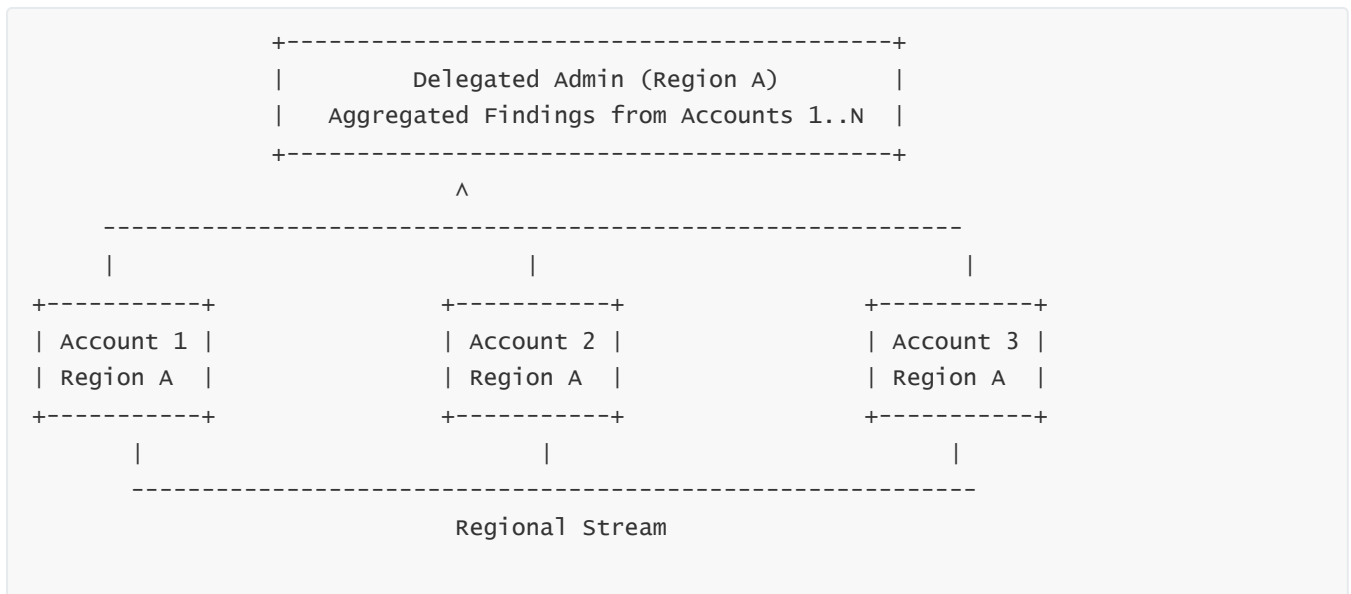
Once the regional admin has findings from all accounts, the Insight Engine is applied across:

- Accounts
- Resource types
- Standards
- Severity levels
- Compliance failures
- Tags
- Organizational Units
- Environments (via custom fields)
- Controls
- Categories

This creates **cross-account analytical summaries**, also known as **Aggregation Insights**.

---

## 3 — The Multi-Region Aggregation Architecture (Visual)



Repeat for every region where Security Hub is enabled.

## 4 — The Core Analytical Dimensions of Aggregation Insights

Aggregation Insights do not simply count findings.

They operate across multiple dimensions simultaneously.

These include:

### 4.1 — Account dimension

Findings grouped by:

- AWS account
- Organizational Unit
- Business unit
- Tag-based ownership

### 4.2 — Region dimension

Identifies regions with:

- High misconfiguration counts
- High threat activity
- Bad security hygiene
- Inconsistent standards enablement

## 4.3 — Resource type dimension

Shows which resource categories are most problematic:

- EC2
- S3
- IAM
- Lambda
- RDS
- VPC
- Containers
- EKS

## 4.4 — Severity distribution

Cross-account severity counts:

- CRITICAL
- HIGH
- MEDIUM
- LOW
- INFORMATIONAL

## 4.5 — Standard and control dimension

Compliance failures grouped by:

- CIS
- PCI
- AWS FSBP
- Custom standards
- Specific control IDs

## 4.6 — Time dimension

Time-based patterns reveal:

- Trending risk
- Spike in vulnerabilities
- Regions drifting at certain times
- Accounts with persistent failures

## 4.7 — Vendor dimension

Grouping by ProductArn reveals:

- Which sources produce the most findings
- Which tools detect the most risk

This multi-dimensionality is what makes Aggregation Insights powerful.

---

## 5 — Data Flow Within Aggregation Insights (Internal Mechanics)

---

Regional SH Stores → Aggregator Store → Insight Query Engine → Aggregated Output

Detailed flow:

### Step 1 — Ingest findings

Each member account ingests findings from AWS + partner tools.

### Step 2 — Sync with admin

Findings flow to the delegated admin.

### Step 3 — Index building

Security Hub builds indexes over:

- severity
- resource type
- account
- compliance state
- control ID
- ProductArn
- timestamps

### Step 4 — Insight evaluation

Insight engine scans indexed data and produces summaries.

### Step 5 — Dashboard rendering

Security Hub console renders aggregated visualizations.

---

## 6 — Organization-Wide Metrics and Dashboards

---

Aggregation Insights surface enterprise-scale metrics such as:

- **Top 10 riskiest accounts**
- **Top 10 failing controls**
- **Distribution of High/Medium/Low findings**
- **Accounts with GuardDuty disabled**
- **Regions with poor compliance**
- **Accounts violating encryption defaults**
- **OWASP / CIS failures across workloads**
- **Misconfigured IAM roles by account**
- **Top threat categories across the org**
- **Trend of critical CVEs in last 30 days**

These dashboards provide Security Hub's strategic view.

---

## 7 — Tag- and OU-Based Aggregation: Governance-Level Insights

---

Security Hub does not natively understand business units or environments.

But organizations use:

- Tags
- Account names
- OU names
- Custom fields

to construct governance-aligned Aggregation Insights, such as:

### “Findings by Environment”

Using tag `Environment=Prod/Dev/Test`.

### “Findings by Business Unit”

Using OUs like `/BU1/Prod`, `/BU1/Dev`.

### “Findings by Application”

Using tags like `App=Billing`.

This creates **organizational context**, which is critical for enterprise governance.

---

## 8 — Cross-Service Correlation Within Aggregation Insights

---

The most powerful capability of aggregation is cross-service correlation.

Security Hub correlates findings across:

- Config
- GuardDuty
- Inspector
- IAM Analyzer
- Macie
- S3 Access Analyzer
- Firewall Manager
- Partner tools

Example correlation:

- Account X has high Inspector CVEs
- Same account has public S3 buckets
- Same account has IAM role trust exposure
- Same account has GuardDuty suspicious API calls

Aggregation Insights reveal **multi-dimensional risk concentration**.

---

## 9 — Aggregation of Compliance Results (Standards)

---

Security Hub aggregates compliance failures across all accounts.

Per standard:

- % compliance score per account
- % compliance per control
- Top failing controls across organization
- Controls failed in >15 accounts
- Compliance drift patterns
- Region-based compliance scores

Example:

### **CIS Control 1.1 – Ensure IAM root access key is not present**

Failed in:

- 3 accounts
- In `us-east-1` only

This informs governance escalations.

---



# 10 — Advanced Aggregation: Severity × Region × Account Matrix

A common analytical structure:

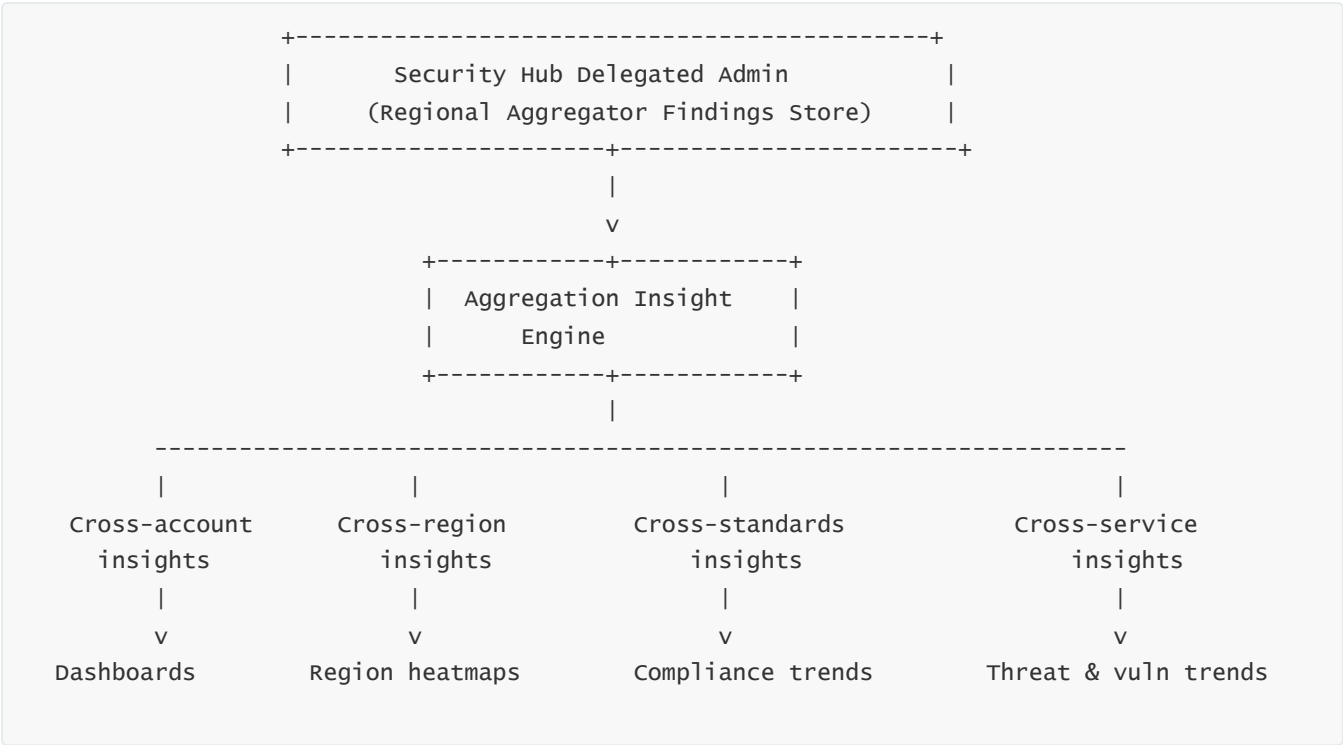
```
Region → Account → Severity Count
```

Example:

```
us-east-1:
  AccountA: 12 HIGH, 3 CRITICAL
  AccountB: 2 HIGH
eu-west-1:
  AccountA: 4 CRITICAL
ap-south-1:
  AccountC: 0 issues
```

This allows global risk distribution analysis.

# 11 — Multi-Layer Diagram of Aggregation Insights Engine



This shows how aggregation becomes the backbone of enterprise posture analysis.

## 12 — Common Aggregation Patterns in Large Enterprises

---

### 12.1 — “Top 10 Riskiest Accounts”

Sorted by weighted severity.

### 12.2 — “Compliance Scoreboard by OU”

Aggregated across controls.

### 12.3 — “Critical CVEs by Account and Region”

Inspector findings grouped cross-region.

### 12.4 — “IAM Misconfigurations Across Org”

IAM Analyzer + Config + Partner CSPM.

### 12.5 — “Public Exposure Violations”

Combines S3, IAM Roles, Lambda URLs.

### 12.6 — “High-Severity Trending Last 30 Days”

Time-based insight across accounts.

---

## 13 — Exporting Aggregated Insights for Enterprise Reporting

---

Enterprises often export aggregated insights to:

- SIEM systems (Splunk, Datadog, QRadar)
- Data lakes (S3 + Glue + Athena)
- PowerBI/Tableau dashboards
- GRC systems
- Compliance audit platforms

Security Hub becomes the **source of normalized security truth**.

---

## 14 — Why Aggregation Insights Are Critical in Real-World Security Ops

---

Aggregation Insights provide:

- CISO-level reporting
- SOC-level prioritization
- Architecture-level governance

- DevOps-level issue tracking
- Enterprise compliance observation
- Organization-wide accountability

Without aggregation, enterprise security visibility becomes fragmented and impossible to operate at scale.

---

## 15 — Mental Model Summary

---

Aggregation Insights transform Security Hub into:

“A federated intelligence layer that consolidates findings from every AWS account and region, correlates them across services, standardizes severity, evaluates compliance, and produces multi-dimensional risk views for operational, tactical, and strategic decision-making.”

This analytical layer is what allows Security Hub to function as a **true organization-wide security posture management system**.

---

# 15 — Remediation Patterns, Automated Response Pipelines, and Enterprise SOAR Integration

---

## 1 — Why Remediation Patterns Are Critical in Security Hub

---

Security Hub is fundamentally a **detection, aggregation, and analysis platform**, but in real-world cloud operations, **detection without remediation is useless**.

Across a large AWS organization:

- Configuration drifts are constant
- Identity misconfigurations reappear
- New CVEs emerge daily
- Threat patterns evolve in real time
- Teams push infrastructure changes continuously
- Accounts, regions, and workloads expand rapidly

A modern cloud security model must be **self-healing**, and Security Hub’s remediation ecosystem provides the backbone that enables AWS environments to move from “reactive ticketing” to **autonomous response**.

Thus, remediation patterns transform Security Hub into a **control plane for enforcing security posture**, not just reporting violations.

---

## 2 — The Remediation Architecture: Security Hub as the Trigger Layer

Think of remediation architecture as a four-stage pipeline:

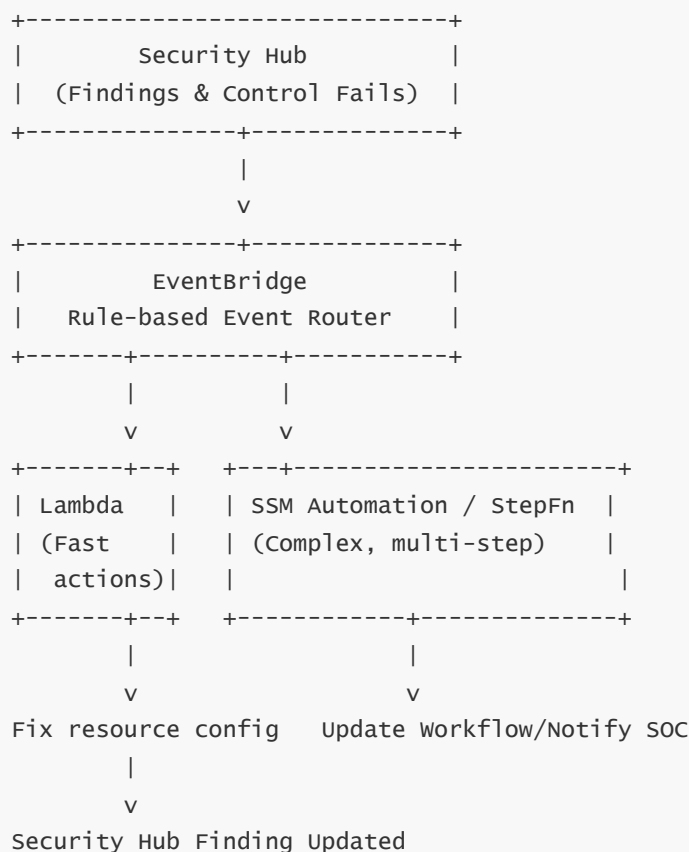
```
Detection → Trigger → Decision → Action
```

Where:

- **Detection** = Security Hub finding
- **Trigger** = EventBridge rule
- **Decision** = Lambda/SSM/Step Functions
- **Action** = Applied fix or workflow update

This entire sequence is event-driven—Security Hub emits structured ASFF events, and your remediation engine consumes them.

## 3 — Multi-Layer Remediation Pipeline Diagram



This pipeline can be simple or extremely sophisticated, depending on enterprise maturity.

## 4 — The Four Primary Remediation Patterns

---

There are four dominant remediation patterns used in Security Hub operational environments:

### 4.1 — Pattern 1: Direct Lambda Remediation (Fast Fixes)

This is the most common pattern, ideal for straightforward configuration fixes.

Flow:

```
Security Hub Finding → EventBridge → Lambda → Fix → Update workflow
```

Examples:

- Remove public S3 bucket access
- Disable public RDS snapshot
- Revoke insecure security group rule
- Enable encryption on Kinesis stream
- Force GuardDuty enablement
- Terminate compromised EC2 instance

Advantages:

- Millisecond latency
- Cheap
- Simple
- Works cross-account (via IAM roles)

---

### 4.2 — Pattern 2: SSM Automation Documents (Complex Fixes)

Used for multi-step or regulated remediation workflows that require:

- Approval
- Escalation
- Rollback
- Multi-resource changes
- Audit logs
- Parameterization

Flow:

```
Security Hub → EventBridge → SSM Automation → Multi-step Fix → Update workflow
```

Examples:

- Patch EC2 instance
- Rebuild AMI
- Enforce encryption across large fleet
- Reset compromised IAM roles
- Regenerate access keys
- Rotate KMS keys or secrets

---

## 4.3 — Pattern 3: Step Functions (Stateful Orchestration)

Used when remediation involves numerous branching paths, conditional logic, or integration with external systems.

Flow:

```
Event → Step Function → External APIs → Remediation → Validation → workflow update
```

Examples:

- Multi-team incident response process
- Container image rebuild → Rescan → Deployment gating
- Automated triage + enrichment + remediation
- Complex IAM or VPC remediation

---

## 4.4 — Pattern 4: SOAR-Based Remediation (Enterprise Security Ops)

SOAR platforms ingest Security Hub events and run their own playbooks.

Platforms include:

- Splunk SOAR
- Cortex XSOAR
- IBM QRadar SOAR
- Rapid7 InsightConnect
- Swimlane
- Tines

Flow:

```
Security Hub → EventBridge or API Export → SOAR → Playbook → Fix → UpdateBackToSH
```

Benefits:

- Human-in-the-loop review

- Enrichment (TI, sandboxing, identity lookups)
  - Ticketing integration
  - Complex orchestrations
- 

## 5 — Deep Internal Mechanics: How Remediation is Triggered

---

### 5.1 — Security Hub publishes events

Every finding update generates:

- “Finding Imported” OR
- “Finding Updated”

These are structured EventBridge events.

### 5.2 — EventBridge filtering

EventBridge matches conditions such as:

- Severity
- Account
- Region
- ProductArn
- Resource type
- Workflow status
- Compliance status
- Control ID

This allows surgical targeting.

### 5.3 — Target invocation

EventBridge routes the event to:

- Lambda
- SQS
- SNS
- Step Functions
- SSM Automation
- Event bus in different account

## 5.4 — Execution

The target executes remediation logic.

## 5.5 — Workflow update

After remediation, the playbook updates:

```
workflow.Status = RESOLVED
```

This completes the cycle.

---

## 6 — Cross-Account Automated Remediation (Enterprise Pattern)

Enterprises usually centralize remediation in a **delegated admin account**, without deploying automation into every member account.

Achieved via:

- IAM assume-role policies
- Step Functions with cross-account tasks
- Lambda with STS role assumption
- SSM Automation executing in member accounts

Flow:

1. Admin receives finding
2. Admin Lambda assumes member account role
3. Executes remediation
4. Updates finding workflow (global sync)

This centralizes governance and reduces duplicated effort.

---

## 7 — Business Logic for Choosing Remediation Path

### Use Lambda when:

- Fix is simple
- Resource impact is isolated
- No approval required
- No rollback needed
- Minimal risk



## Use SSM Automation when:

- Fix involves OS-level actions
- Multi-step workflow
- Needs auditing or approval
- Affects multiple resources

## Use Step Functions when:

- Many conditional paths
- Must coordinate API calls
- Must involve multiple AWS services
- Requires validation after fix

## Use SOAR when:

- Enrichment with threat intel required
- Manual review included
- External ticketing automation needed
- IR team involved

---

## 8 — Automated Remediation for Compliance Findings

Security Hub is often used to enforce organizational guardrails.

Example:

Control “S3.1 - Ensure public access block enabled.”

Remediation pipeline:

```
NEW finding → EventBridge → Lambda → Block public access → Update workflow → Control  
reevaluates → PASSED
```

Compliance controls are highly automatable because they relate to predictable misconfigurations.

---

## 9 — Automated Remediation for Threat Findings

Threat findings require more careful handling:

- GuardDuty: compromised credentials
- Inspector: malware
- CWPP: suspicious runtime behavior

Automation pattern:

1. Quarantine resource
2. Trigger IR workflow
3. Collect forensics (SSM run command)
4. Revoke temporary credentials
5. Block malicious IP in NACL/WAF
6. Notify SOC
7. Update workflow state

These workflows vary depending on risk tolerance.

---

## 10 — Remediation in Regulated Environments (PCI, HIPAA, RBI, etc.)

---

Highly regulated organizations require:

- Evidence collection
- Approval before remediation
- Change management integration
- Two-person review
- Timestamped audit logs

SSM Automation and Step Functions support these requirements natively:

```
Event → Pre-approval → Execution → Validation → Ticket update → SH workflow update
```

This ensures compliance with regulatory audit expectations.

---

## 11 — Integration with Code Pipelines and CI/CD for Preventative Fixes

---

Findings often indicate issues that originate in infrastructure-as-code or CI/CD pipelines.

Organizations integrate Security Hub with:

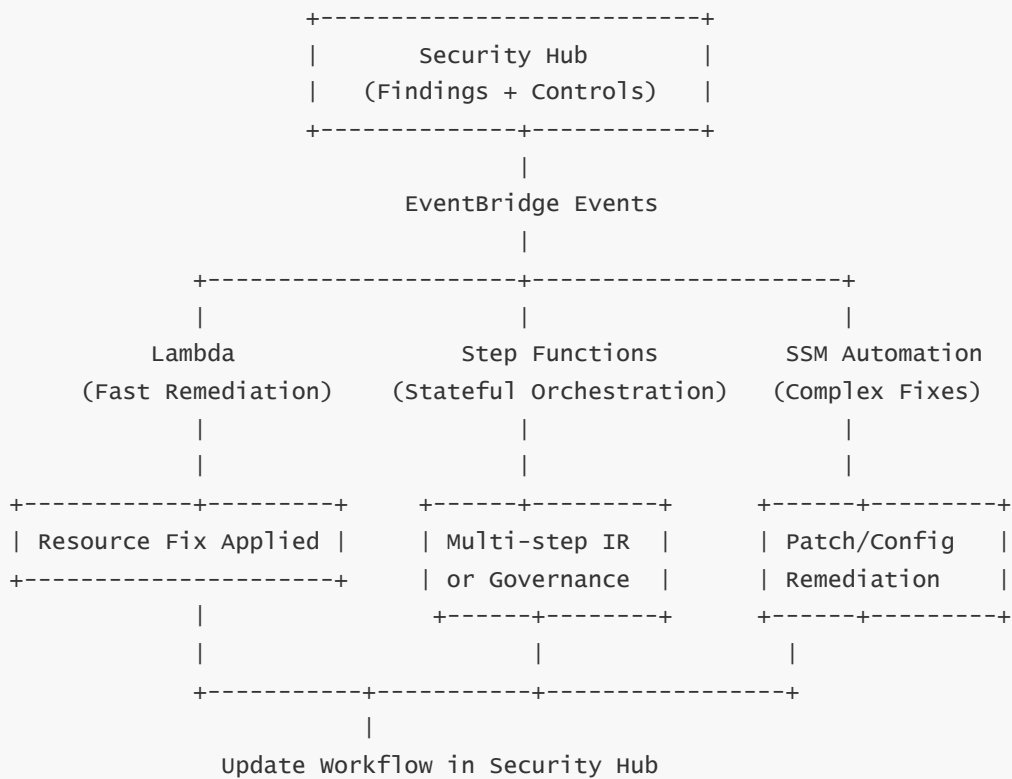
- AWS CodePipeline
- GitHub Actions
- GitLab CI
- Jenkins
- Harness
- Spinnaker

Workflow:

Security Hub Finding → EventBridge → Pipeline block → Developer notified → Fix in IaC → Redeploy

This shifts remediation “left,” preventing recurring issues.

## 12 — Multi-Layer Remediation Orchestration Diagram



This diagram shows the complete remediation ecosystem.

## 13 — Remediation Playbooks: The Master Catalog

Enterprises maintain playbooks such as:

### Identity & Access:

- Remove public access from IAM roles
- Force MFA
- Disable access keys
- Quarantine compromised IAM entities

## Storage:

- Patch S3 bucket encryption
- Fix RDS snapshot permissions
- Enforce KMS on EBS volumes

## Compute:

- Stop compromised EC2 instances
- Rebuild AMIs
- Apply missing patches

## Network:

- Remove risky SG rules
- Update NACLs
- Modify route tables

## Threat Response:

- Block malicious IPs
- Revoke session tokens
- Disk snapshot for forensic analysis

These playbooks are implemented using Lambda, SSM, Step Functions, or SOAR.

---

## 14 — Remediation Governance and Escalation Models

---

Enterprises structure remediation workflows as:

### Tier 0 — Fully Automated

Low-risk, high-volume fixes.

### Tier 1 — Partially Automated

Fix prepared automatically but requires approval.

### Tier 2 — Manual Review

Used for sensitive resources.

### Tier 3 — Incident Response

SOC + IR + Security Engineering involvement.

Security Hub workflows + EventBridge pipelines dictate escalation.

---

## 15 — Mental Model Summary

---

Summarizing the complete remediation patterns:

Security Hub is the detection and orchestration brain.  
EventBridge is the routing engine.  
Lambda/SSM/Step Functions/SOAR are the hands that apply the fix.  
Workflow states record and synchronize the lifecycle.  
Together, they form a distributed self-healing security system.

This is the foundation of modern cloud security operations at enterprise scale.

---

## 16 — Security Hub Dashboards, Insight Visualizations, and Enterprise Reporting Architecture

---

### 1 — Why Dashboards and Visualizations Are Critical for Security Hub

---

Security Hub ingests **massive volumes of heterogeneous findings** across AWS services, partner tools, compliance engines, threat detectors, and vulnerability scanners.

To make this data **actionable**, Security Hub provides a layered visualization framework that converts raw ASFF findings into:

- Executive dashboards
- SOC triage views
- Compliance scorecards
- Regional risk heatmaps
- Account-level security posture summaries
- Time-series trend charts
- Service-specific analysis layers
- Custom Insight dashboards mapped to business structure

These dashboards are essential because raw JSON-based findings cannot provide:

- Comparative risk views
- Prioritization maps
- Compliance summaries
- Trend analysis
- Regional/account-level security posture
- Business-unit segmentation

Dashboards and visualization layers are what turn Security Hub from a collection of alerts into a **complete security posture management platform (CSPM)**.

---

## 2 — The Three-Layer Dashboard Architecture in Security Hub

---

Security Hub dashboards are built on a **three-layer architecture**:

1. Finding Layer (raw ASFF)
2. Insight Layer (query + aggregation)
3. Visualization Layer (dashboards, trend charts)

Let's examine each layer deeply.

---

## 3 — Layer 1: Finding Layer (Raw Data Source)

---

Every dashboard eventually pulls its data from normalized ASFF findings stored in regional Security Hub stores.

### The Finding Layer includes:

- Raw findings
- Severity
- Compliance status
- Timestamps
- Resource metadata
- Control IDs
- ProductArns
- Workflow states
- Account IDs
- Regions
- Compliance mapping

Dashboards never operate directly on raw JSON.

Instead, the Insight engine transforms the raw data into structured subsets.

---

## 4 — Layer 2: Insight Layer (Query + Grouping Engine)

---

Insights form the **query execution layer** of the dashboard system.

Key elements of the Insight layer:

- Filter (scope)
- Group-by (dimension)

- Sorting
- Metrics (counts, compliance ratios)
- Derived fields (standards, controls, severity groups)

This layer supports:

- Custom Insights
- Managed Insights
- Organization-wide Insights
- Cross-region and cross-account aggregation

Insights are not precomputed static objects; they are **dynamically evaluated live** across indexed findings.

---

## 5 — Layer 3: Visualization Layer (Dashboards)

---

Dashboards are the **rendered surface** of Insights.

Security Hub includes:

### 5.1 — Posture Overview Dashboard

Shows:

- Total findings
- Current open CRITICAL/HIGH findings
- Distribution by severity
- Latest compliance failures

### 5.2 — Standards Dashboard

Shows:

- CIS
- PCI
- AWS FSBP
- Custom standards
- Control pass/fail ratios
- Compliance percentage per standard
- Historical compliance badges

### 5.3 — Insights Dashboard

Displays:

- Top accounts with most findings
- Top failing controls
- Resource-type breakdown

- Severity trends
- Threat-type distributions
- Volume of findings per service (e.g., Inspector vs GuardDuty)

## 5.4 — Account Aggregation Dashboard

Shows:

- Per-account severity distribution
- Accounts with recurring failures
- Accounts missing key controls (e.g., GuardDuty disabled)
- Per-account compliance posture

Dashboards become the central visibility surface for cloud security leadership.

---

## 6 — Dashboard Rendering Pipeline (Internal Mechanics)

ASFF Store → Indexing → Insight Engine Query → Aggregation Results → Visualization Renderer

Deep flow:

### 6.1 — ASFF Store Query

Insight engine pulls data based on filters.

### 6.2 — Index-based scanning

Indexes for severity, account, region, control, time windows accelerate retrieval.

### 6.3 — Group-by computation

Counts per:

- account
- severity
- service
- standards
- controls
- region
- resource type



## 6.4 — Sorting and ranking

Top-N applied based on metrics.

## 6.5 — Visualization rendering

Console renders data as:

- bar graphs
- trending lines
- scorecards
- grouped tables
- compliance diamonds
- heatmaps

---

## 7 — Cross-Region + Cross-Account Dashboard Consolidation

Since Security Hub operates regionally, dashboards reflect aggregated data per region via the delegated admin.

Workflow:

Member Accounts → Regional Finding Stores → Regional Admin → Dashboard Aggregation

Thus, dashboards show consolidated views for all accounts in the region, but must be accessed **per region**.

Enterprises then export or replicate these dashboards centrally.

---

## 8 — The Core Built-In Dashboards Explained (Deep Detail)

### 8.1 — Posture Overview Dashboard

Provides a cross-service risk summary:

- Total findings (active)
- Severity distribution (pie + bar)
- New findings in last 24 hours
- Account-level heat tiles
- Trends compared to previous intervals

Key benefit:

This is the CISO-level “health check” of the cloud environment.

---

## 8.2 — Security Standards Dashboard

This is the compliance command center.

Shows:

- Per-standard compliance score
- Per-control pass/fail
- Failed resource count
- Remediation progress per control
- Historical compliance improvements
- OU/account-based filtering
- Region-based control failures

This dashboard is essential for:

- PCI audits
  - CIS posture tracking
  - SOC2 or ISO27001 readiness
  - Organizational governance
- 

## 8.3 — Insights Dashboard

This is the SOC / security architect's analytical workspace.

Contains:

- Top failing controls
- Top misconfigured resource types
- Top severity counts
- Region distribution
- Source distribution (Inspector, GuardDuty, etc.)
- Trend lines for high-severity over time
- Findings grouped by environment (via tags)

This dashboard drives **action**, not just reporting.

---

## 9 — Dashboard Personalization Using Custom Insights

Organizations define custom dashboards by assembling custom Insights.

Examples:

## 9.1 — Application-Level Dashboards

Group findings by tag: `App=Billing`, `App=Payments`.

## 9.2 — Environment-Level Dashboards

Group by tags: `Environment=Prod`, `Dev`, `QA`.

## 9.3 — OU-Level Dashboards

Filter by OU-based account lists.

## 9.4 — Compliance Team Dashboards

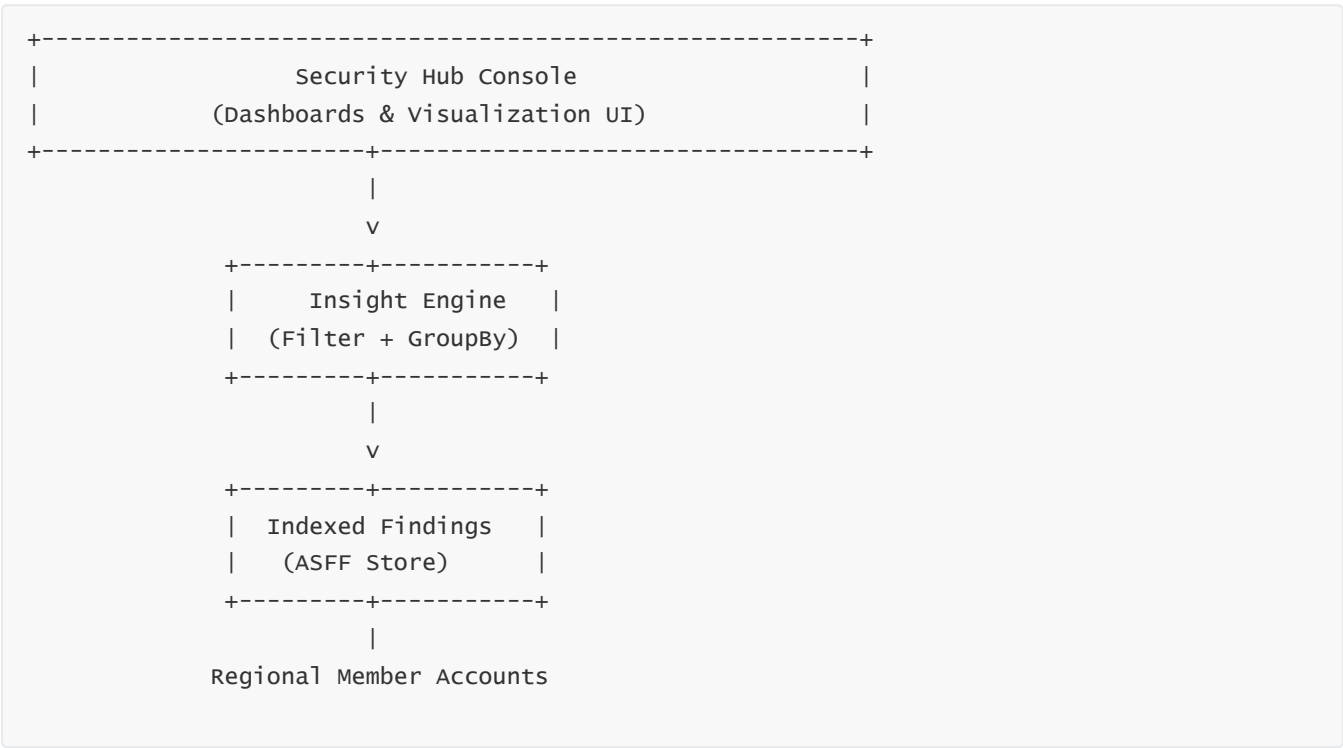
- Controls failing across critical workloads
- Repeat offenders
- Controls grouped by severity impact

## 9.5 — Threat Intelligence Dashboards

- GuardDuty findings by threat type
- Threat actor category distribution
- Network-based detections per VPC

Custom Insights provide infinite flexibility when constructing dashboards.

# 10 — Multi-Layer Internal Architecture Diagram for Dashboarding



This diagram represents the entire visualization pipeline.

---

## 11 — Exporting Dashboards for Enterprise Reporting

---

Dashboards often need to be exported to systems like:

- PowerBI
- Tableau
- Grafana
- Splunk
- Elastic/Kibana
- GRC systems
- Executive reporting portals

Export methods:

### 11.1 — EventBridge + Lambda export

Stream findings into S3 → Athena → BI tools.

### 11.2 — API-driven extraction

Use `GetFindings`, `GetInsightsResults`, etc.

### 11.3 — Security Lake integration

Feed findings into Amazon Security Lake → central analytics.

### 11.4 — Partner connectors

SIEM/SOAR connectors for dashboards.

Enterprises typically build unified “Enterprise Posture Dashboards” on top of these pipelines.

---

## 12 — Time-Series Trends and Historical Posture Visualization

---

Security Hub visualizations show trends of:

- HIGH/CRITICAL findings over time
- New vs resolved findings
- Daily/weekly compliance drift
- Region-based drift timelines
- Control-by-control improvement trends

Trend analysis is one of the most valuable aspects of dashboarding because it enables:

- Predictive insights

- Early detection of security regressions
  - Governance enforcement
  - KPI & OKR tracking for security teams
- 

## 13 — Enterprise Use Cases for Security Hub Dashboards

---

### 13.1 — Executive-level reporting

- Heatmaps
- KPI trendlines
- High-risk account lists
- Compliance percentages
- Severity distribution

### 13.2 — SOC-level dashboards

- Threat alert distribution
- High-severity queue
- Workflow monitoring
- Remediation SLA tracking

### 13.3 — Governance dashboards

- Compliance drift
- Control failure patterns
- OU or BU compliance scoring

### 13.4 — DevOps dashboards

- Findings per application
- Findings per deployment
- Service-level hygiene

Dashboards unify these views into a shared security truth.

---

## 14 — Dashboards as Inputs to Automation Pipelines

---

Dashboards also influence automated workflows:

Example:

- Insight shows a trending increase in public S3 buckets
- Automation system updates policies
- CI/CD injection prevents future occurrences

Dashboards → Insights → Policy → Automation

This loop enables **continuous posture improvement**.

---

## 15 — Mental Model Summary

---

The entire dashboard ecosystem can be summarized as:

“Dashboards are the visualization layer built on top of Insight queries, which themselves sit over indexed ASFF findings. Together, they provide tactical, operational, and strategic security visibility across accounts, regions, services, and compliance frameworks.”

Dashboards are the **front-end of enterprise posture intelligence**, enabling every tier of the organization to understand and improve cloud security.

---

## 17 — Integration with AWS Security Lake, SIEM Platforms, and Enterprise Data Pipelines

---

### 1 — Why Security Hub Needs Integration with Security Lake and SIEM Systems

---

Security Hub is the **centralized posture, compliance, and finding aggregation engine** for AWS security, but it is *not* designed to be:

- a long-term data lake
- a full forensic repository
- a correlation engine for multi-cloud/on-prem logs
- a deep analytics platform
- an enterprise-wide SIEM replacement

Most enterprises require unified visibility that spans:

- AWS (Security Hub, GuardDuty, Inspector, CloudTrail, VPC Flow Logs)
- On-premises environments
- Multi-cloud platforms (Azure, GCP)
- Endpoint telemetry
- Identity logs (AD, Okta, IAM)
- Application logs
- Network sensors
- Threat intelligence feeds

To achieve this unified visibility, Security Hub integrates with:

- **Amazon Security Lake** (native AWS analytics lake)

- **SIEM platforms** (Splunk, QRadar, Elastic, LogRhythm, ArcSight, Chronicle)
- **EDR/XDR platforms** (CrowdStrike, SentinelOne, Palo Alto Cortex)
- **SOAR systems**
- **GRC/Compliance tools**

These integrations allow organizations to treat Security Hub as a core **normalized security data source** while offloading deep analytics to tools built for long retention and correlation.

---

## 2 — The Three Major External Integration Categories

---

Security Hub integrates with external platforms through three main channels:

1. Amazon Security Lake Integration
2. SIEM Integration (native + partner)
3. Custom Data Pipelines (S3, EventBridge, API, Lambda)

Each serves different architectural purposes.

---

## 3 — Integration Category 1: Amazon Security Lake (Deepest Native Integration)

---

### 3.1 — What Security Lake Is

Security Lake is a centralized, cross-account, cross-region **security data store** built around the **Open Cybersecurity Schema Framework (OCSF)**.

It aggregates and normalizes logs and findings from:

- Security Hub
- GuardDuty
- Inspector
- CloudTrail
- VPC Flow Logs
- S3 Access Logs
- Application logs
- Custom logs

Security Lake provides:

- Long-term storage
- Normalized data structure
- Querying with Athena
- Integration with SIEM/Data Lakes

- Unified multi-source correlation

This makes it the default choice for deep analytics.

## 3.2 — How Security Hub Integrates with Security Lake

Security Hub findings are automatically converted into **OCSF format** and delivered into Security Lake partitions by region/account.

Flow:

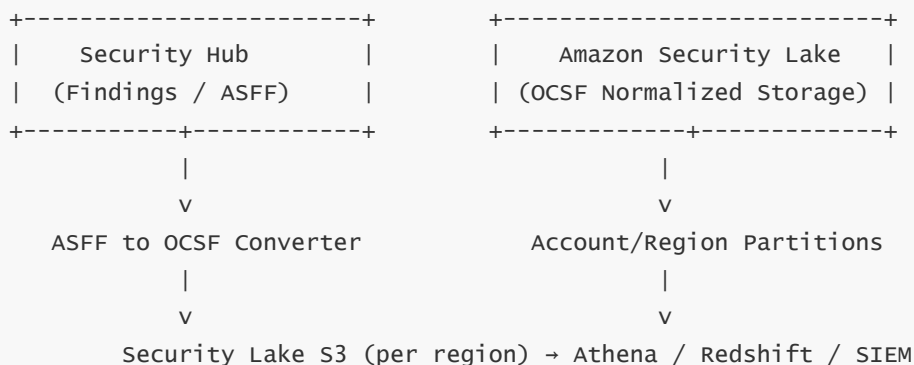
Security Hub Findings → OCSF Transformation → Security Lake S3 → Lake Formation Governance → Athena / SIEM

This enables:

- SQL-based investigation
- Multi-year trend evaluation
- Cross-service correlation
- Federated analytics

Security Lake becomes the **data warehouse** for Security Hub outputs.

## 3.3 — Security Hub → Security Lake Architecture Diagram



This pipeline forms the backbone of AWS-native security analytics.

## 4 — Integration Category 2: SIEM Platforms (Enterprise SOC Integration)

Security Hub integrates with SIEMs in two primary ways:



- A. Direct Integration via Partner Connectors
- B. Indirect Integration via Security Lake or S3 Export

## 4.1 — Direct SIEM Integrations

Many SIEMs have AWS-native or partner connectors for Security Hub:

- Splunk Add-on for AWS
- IBM QRadar AWS Security Hub DSM
- Microsoft Sentinel AWS Security Hub connector
- Google Chronicle HTTP/S3 collector
- Datadog Security Hub Integration
- Elastic Agent Integration

This approach streams findings directly into the SIEM.

## 4.2 — SIEM Integration Architecture

```
Security Hub → EventBridge → Lambda Transform (optional) → SIEM Ingest API
```

or

```
Security Hub → S3 (via exporter) → SIEM S3 collector
```

## 4.3 — What SIEMs Use Security Hub For

SIEMs typically use Security Hub data for:

- Alert correlation (across AWS + on-prem)
- Incident triage
- Dashboards
- Threat detection rules
- Compliance monitoring
- Historical trend reporting
- SOC enrichment pipelines

SIEMs treat Security Hub as a *structured high-value alert feed*.

---

## 5 — Integration Category 3: Custom Data Pipelines (S3, EventBridge, API)

---

Advanced enterprises build custom pipelines:

## 5.1 — S3-Based Export Pipelines

Flow:

```
Security Hub → EventBridge → Lambda → S3 → Athena/Glue/EMR/ETL pipelines
```

Used for:

- Long-term archiving
- Data science workloads
- Forensic pipelines
- Machine learning models
- Custom dashboards

## 5.2 — EventBridge Streaming Pipelines

Flow:

```
Security Hub → EventBridge → Kinesis / Firehose → SIEM / Data Lake
```

Used for high-speed multi-stream architectures.

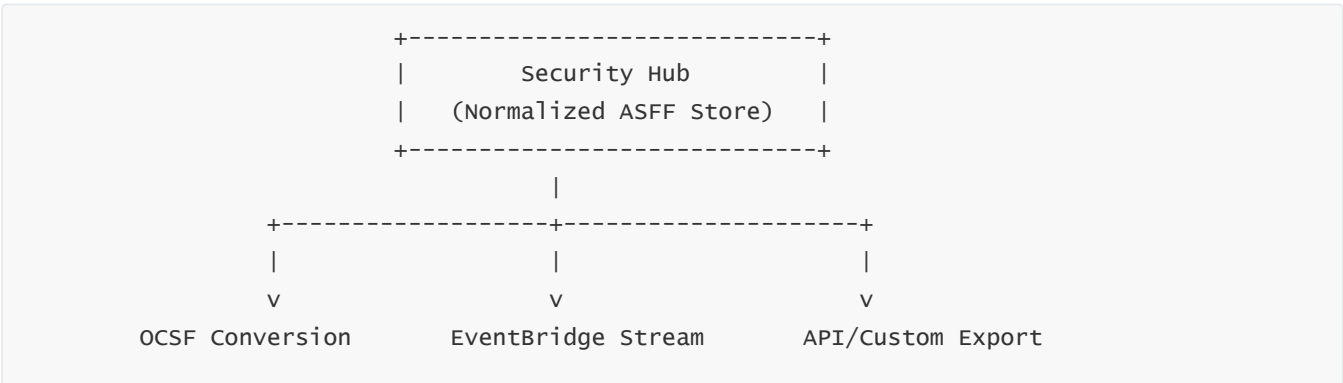
## 5.3 — Direct API Extraction

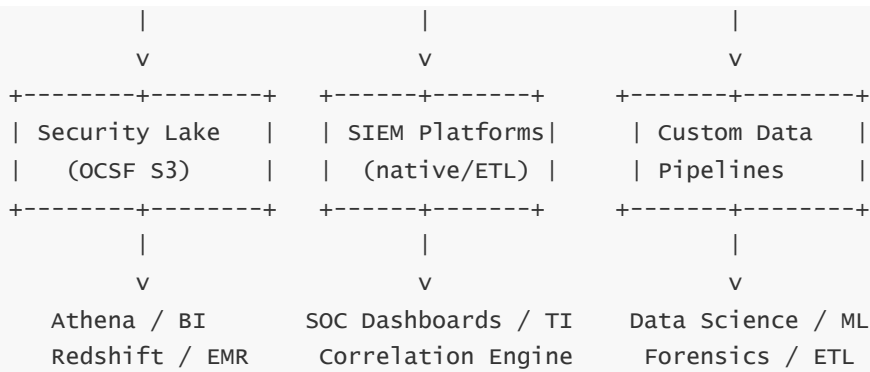
Using `GetFindings`, `GetInsights`, `BatchImportFindings`, etc.

Used for:

- Custom dashboards
- Case management platforms
- Policy engines
- Cloud governance tools

# 6 — Deep Integration Flow: Unified Diagram Covering All Paths





This is the full multi-path integration architecture used in enterprise SOC environments.

## 7 — How Security Lake Complements Security Hub (Internal Understanding)

Security Hub provides:

- Normalized findings
- Compliance results
- Severity scoring
- Insight engine
- Governance dashboards

Security Lake provides:

- Long-term storage for logs
- Cross-service correlation
- SQL-based investigation
- Multi-source analytics
- Scalability for massive workloads
- ML training data

Together:

- Security Hub ≈ "What happened?"
- Security Lake ≈ "Why did it happen? How often? Across what systems?"

Security Hub → Security Lake → SIEM → SOAR

This is the modern enterprise security pipeline.

## 8 — Data Retention Architecture

---

### 8.1 — Security Hub Retention

Security Hub does **not** store findings forever.

It stores the *active state* of findings.

### 8.2 — Security Lake Retention

Security Lake stores years of historical data.

### 8.3 — SIEM Retention

SIEMs store whatever retention the enterprise decides (90 days → years).

Thus, Security Hub is the **live posture**, not the historical archive.

---

## 9 — Multi-Cloud Integration Through SIEM and Security Lake

---

Security Hub does not natively ingest:

- Azure Defender findings
- GCP Security Command Center
- On-prem EDR logs
- Firewall logs
- DLP engines
- NDR platforms

But SIEM/Security Lake correlate these seamlessly.

### Example:

```
Security Hub (AWS Findings)
Azure Defender (Azure Findings)
CrowdStrike (EDR)
Palo Alto Firewall (NDR)
CloudTrail, VPC Flow Logs
```

Together form a multi-cloud threat graph.

---

## 10 — Integration With GRC and Compliance Tools

---

Security Hub integrates with GRC platforms like:

- ServiceNow GRC

- Archer
- Drata
- Secureframe
- Tugboat Logic
- AuditBoard

Flow:

Security Hub → API Export → GRC Tool → Compliance Mapping

GRC systems use Security Hub controls to populate compliance posture automatically.

---

## 11 — High-Fidelity Enterprise Reporting Using Integration Pipelines

---

Companies build reporting layers:

- SOC dashboards
- Quarterly risk reports
- Board-level summaries
- PCI DSS / SOX / HIPAA evidence reports
- SLA breach tracking
- Benchmark comparisons
- Trendlines for cloud maturity

Security Hub becomes the “active truth,” while the integration systems become the “historical truth.”

---

## 12 — Mental Model Summary

---

You can summarize this entire integration ecosystem as:

Security Hub is the unified AWS security alert brain.  
Security Lake is the long-term, normalized security data warehouse.  
SIEM is the enterprise-wide correlation and detection engine.  
SOAR is the automated response engine.  
Custom pipelines interconnect all of them.

Security Hub acts as the **primary source of structured security findings**, while its integrations form a complete ecosystem for analytics, compliance, detection, forensics, and automation.

---

## 17 — Integration with AWS Security Lake, SIEM Platforms, and Enterprise Data Pipelines

---

# 1 — Why Security Hub Needs Integration with Security Lake and SIEM Systems

---

Security Hub is the **centralized posture, compliance, and finding aggregation engine** for AWS security, but it is *not* designed to be:

- a long-term data lake
- a full forensic repository
- a correlation engine for multi-cloud/on-prem logs
- a deep analytics platform
- an enterprise-wide SIEM replacement

Most enterprises require unified visibility that spans:

- AWS (Security Hub, GuardDuty, Inspector, CloudTrail, VPC Flow Logs)
- On-premises environments
- Multi-cloud platforms (Azure, GCP)
- Endpoint telemetry
- Identity logs (AD, Okta, IAM)
- Application logs
- Network sensors
- Threat intelligence feeds

To achieve this unified visibility, Security Hub integrates with:

- **Amazon Security Lake** (native AWS analytics lake)
- **SIEM platforms** (Splunk, QRadar, Elastic, LogRhythm, ArcSight, Chronicle)
- **EDR/XDR platforms** (CrowdStrike, SentinelOne, Palo Alto Cortex)
- **SOAR systems**
- **GRC/Compliance tools**

These integrations allow organizations to treat Security Hub as a core **normalized security data source** while offloading deep analytics to tools built for long retention and correlation.

---

## 2 — The Three Major External Integration Categories

---

Security Hub integrates with external platforms through three main channels:

1. Amazon Security Lake Integration
2. SIEM Integration (native + partner)
3. Custom Data Pipelines (S3, EventBridge, API, Lambda)

Each serves different architectural purposes.

---

## 3 — Integration Category 1: Amazon Security Lake (Deepest Native Integration)

---

### 3.1 — What Security Lake Is

Security Lake is a centralized, cross-account, cross-region **security data store** built around the **Open Cybersecurity Schema Framework (OCSF)**.

It aggregates and normalizes logs and findings from:

- Security Hub
- GuardDuty
- Inspector
- CloudTrail
- VPC Flow Logs
- S3 Access Logs
- Application logs
- Custom logs

Security Lake provides:

- Long-term storage
- Normalized data structure
- Querying with Athena
- Integration with SIEM/Data Lakes
- Unified multi-source correlation

This makes it the default choice for deep analytics.

---

### 3.2 — How Security Hub Integrates with Security Lake

Security Hub findings are automatically converted into **OCSF format** and delivered into Security Lake partitions by region/account.

Flow:

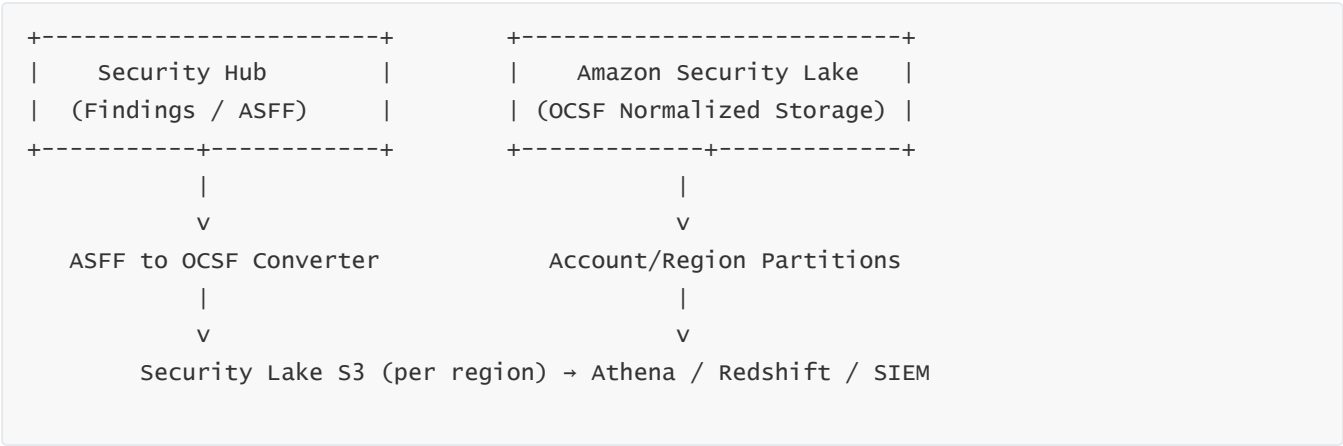
Security Hub Findings → OCSF Transformation → Security Lake S3 → Lake Formation Governance → Athena / SIEM

This enables:

- SQL-based investigation
- Multi-year trend evaluation
- Cross-service correlation
- Federated analytics

Security Lake becomes the **data warehouse** for Security Hub outputs.

### 3.3 — Security Hub → Security Lake Architecture Diagram



This pipeline forms the backbone of AWS-native security analytics.

## 4 — Integration Category 2: SIEM Platforms (Enterprise SOC Integration)

Security Hub integrates with SIEMs in two primary ways:

- A. Direct Integration via Partner Connectors
- B. Indirect Integration via Security Lake or S3 Export

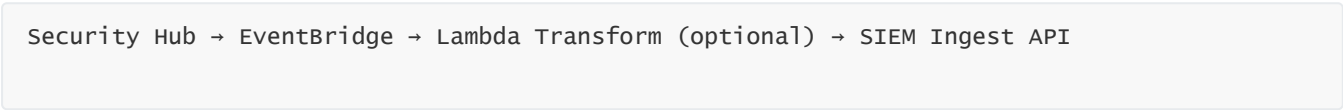
### 4.1 — Direct SIEM Integrations

Many SIEMs have AWS-native or partner connectors for Security Hub:

- Splunk Add-on for AWS
- IBM QRadar AWS Security Hub DSM
- Microsoft Sentinel AWS Security Hub connector
- Google Chronicle HTTP/S3 collector
- Datadog Security Hub Integration
- Elastic Agent Integration

This approach streams findings directly into the SIEM.

### 4.2 — SIEM Integration Architecture



or



Security Hub → S3 (via exporter) → SIEM S3 collector

## 4.3 — What SIEMs Use Security Hub For

SIEMs typically use Security Hub data for:

- Alert correlation (across AWS + on-prem)
- Incident triage
- Dashboards
- Threat detection rules
- Compliance monitoring
- Historical trend reporting
- SOC enrichment pipelines

SIEMs treat Security Hub as a *structured high-value alert feed*.

---

## 5 — Integration Category 3: Custom Data Pipelines (S3, EventBridge, API)

Advanced enterprises build custom pipelines:

### 5.1 — S3-Based Export Pipelines

Flow:

Security Hub → EventBridge → Lambda → S3 → Athena/Glue/EMR/ETL pipelines

Used for:

- Long-term archiving
- Data science workloads
- Forensic pipelines
- Machine learning models
- Custom dashboards

### 5.2 — EventBridge Streaming Pipelines

Flow:

Security Hub → EventBridge → Kinesis / Firehose → SIEM / Data Lake

Used for high-speed multi-stream architectures.

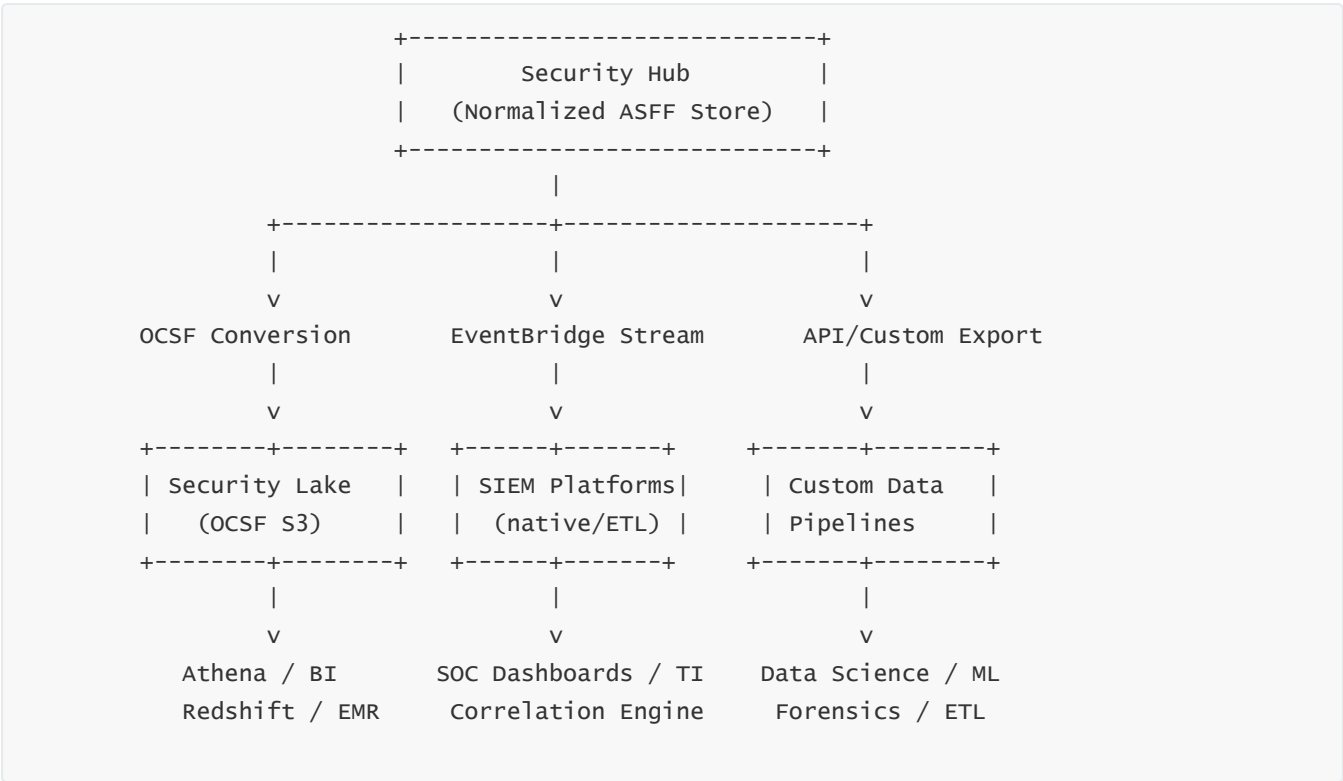
### 5.3 — Direct API Extraction

Using `GetFindings`, `GetInsights`, `BatchImportFindings`, etc.

Used for:

- Custom dashboards
- Case management platforms
- Policy engines
- Cloud governance tools

## 6 — Deep Integration Flow: Unified Diagram Covering All Paths



This is the full multi-path integration architecture used in enterprise SOC environments.

## 7 — How Security Lake Complements Security Hub (Internal Understanding)

Security Hub provides:

- Normalized findings
- Compliance results
- Severity scoring
- Insight engine
- Governance dashboards

Security Lake provides:

- Long-term storage for logs
- Cross-service correlation
- SQL-based investigation
- Multi-source analytics
- Scalability for massive workloads
- ML training data

Together:

- Security Hub ≈ "What happened?"
- Security Lake ≈ "Why did it happen? How often? Across what systems?"

Security Hub → Security Lake → SIEM → SOAR

This is the modern enterprise security pipeline.

---

## 8 — Data Retention Architecture

---

### 8.1 — Security Hub Retention

Security Hub does **not** store findings forever.

It stores the *active state* of findings.

### 8.2 — Security Lake Retention

Security Lake stores years of historical data.

### 8.3 — SIEM Retention

SIEMs store whatever retention the enterprise decides (90 days → years).

Thus, Security Hub is the **live posture**, not the historical archive.

---

## 9 — Multi-Cloud Integration Through SIEM and Security Lake

---

Security Hub does not natively ingest:

- Azure Defender findings
- GCP Security Command Center
- On-prem EDR logs
- Firewall logs
- DLP engines
- NDR platforms

But SIEM/Security Lake correlate these seamlessly.

## Example:

Security Hub (AWS Findings)  
Azure Defender (Azure Findings)  
CrowdStrike (EDR)  
Palo Alto Firewall (NDR)  
CloudTrail, VPC Flow Logs

Together form a multi-cloud threat graph.

---

## 10 — Integration With GRC and Compliance Tools

Security Hub integrates with GRC platforms like:

- ServiceNow GRC
- Archer
- Drata
- Secureframe
- Tugboat Logic
- AuditBoard

Flow:

Security Hub → API Export → GRC Tool → Compliance Mapping

GRC systems use Security Hub controls to populate compliance posture automatically.

---

## 11 — High-Fidelity Enterprise Reporting Using Integration Pipelines

Companies build reporting layers:

- SOC dashboards
- Quarterly risk reports
- Board-level summaries
- PCI DSS / SOX / HIPAA evidence reports
- SLA breach tracking
- Benchmark comparisons
- Trendlines for cloud maturity

Security Hub becomes the “active truth,” while the integration systems become the “historical truth.”

---

## 12 — Mental Model Summary

---

You can summarize this entire integration ecosystem as:

Security Hub is the unified AWS security alert brain.  
Security Lake is the long-term, normalized security data warehouse.  
SIEM is the enterprise-wide correlation and detection engine.  
SOAR is the automated response engine.  
Custom pipelines interconnect all of them.

Security Hub acts as the **primary source of structured security findings**, while its integrations form a complete ecosystem for analytics, compliance, detection, forensics, and automation.

---

## 18 — Multi-Region Deployment Strategy, Cross-Region Replication Model, and Global Security Posture Consistency in Security Hub

---

### 1 — Why Multi-Region Strategy Matters for Security Hub

---

Security Hub is **inherently regional**, meaning:

- Findings are stored **per region**
- Compliance standards run **per region**
- Controls evaluate resources **per region**
- Dashboards visualize data **per region**
- Admin delegation is established **per region**

Enterprises operating across 10–20 AWS regions must ensure **consistent security posture**, centralized administration, and uniform governance.

This is complex because AWS resources, threat surfaces, workloads, and compliance risks often differ across regions.

Multi-region strategy answers critical questions:

- How do we maintain one global security posture across regions?
- How do we avoid region-by-region drift in compliance?
- How do we see a consistent risk overview across all regions?
- How do we architect delegated admin roles for many regions?
- How do we avoid doubling/tripling Security Hub costs unnecessarily?
- How do we ensure controls remain uniformly enabled?
- How do we manage global automation pipelines?

A large-scale organization cannot rely on console clicks; multi-region strategy must be **systematized and automated**.

---

## 2 — The Multi-Region Architecture Model of Security Hub

---

The architecture consists of three layers:

```
Regional Member Accounts
↓
Regional Admin Account (per region)
↓
Central Org-Level Visibility (aggregated manually)
```

Security Hub does **not** perform cross-region aggregation automatically.

Organizations must build the global aggregation layer themselves.

Let us break these layers down.

---

## 3 — Layer 1: Regional Member Account Architecture

---

Each AWS account in each region runs:

- Local Security Hub findings store
- Local compliance control evaluations
- Local EventBridge events for findings
- Local workflows and insights
- Local data ingestion from AWS services

This is by design — AWS ensures that compliance and detection remain local to the region's resources and data.

---

## 4 — Layer 2: Delegated Admin per Region

---

Security Hub allows you to designate **one delegated admin per region** for your organization.

This regional delegated admin receives all findings from all member accounts in that region.

For example:

```
Region: ap-south-1
Admin: SecurityAccount
Members: 50 production accounts
```

The delegated admin becomes the **regional view** for posture.

Without enabling the admin account per region, you will not get consolidated analysis.

## 5 — Layer 3: Central Org-Level Aggregation (Customer-Built)

Security Hub does **not** offer global aggregation across regions.

Enterprises build a custom layer such as:

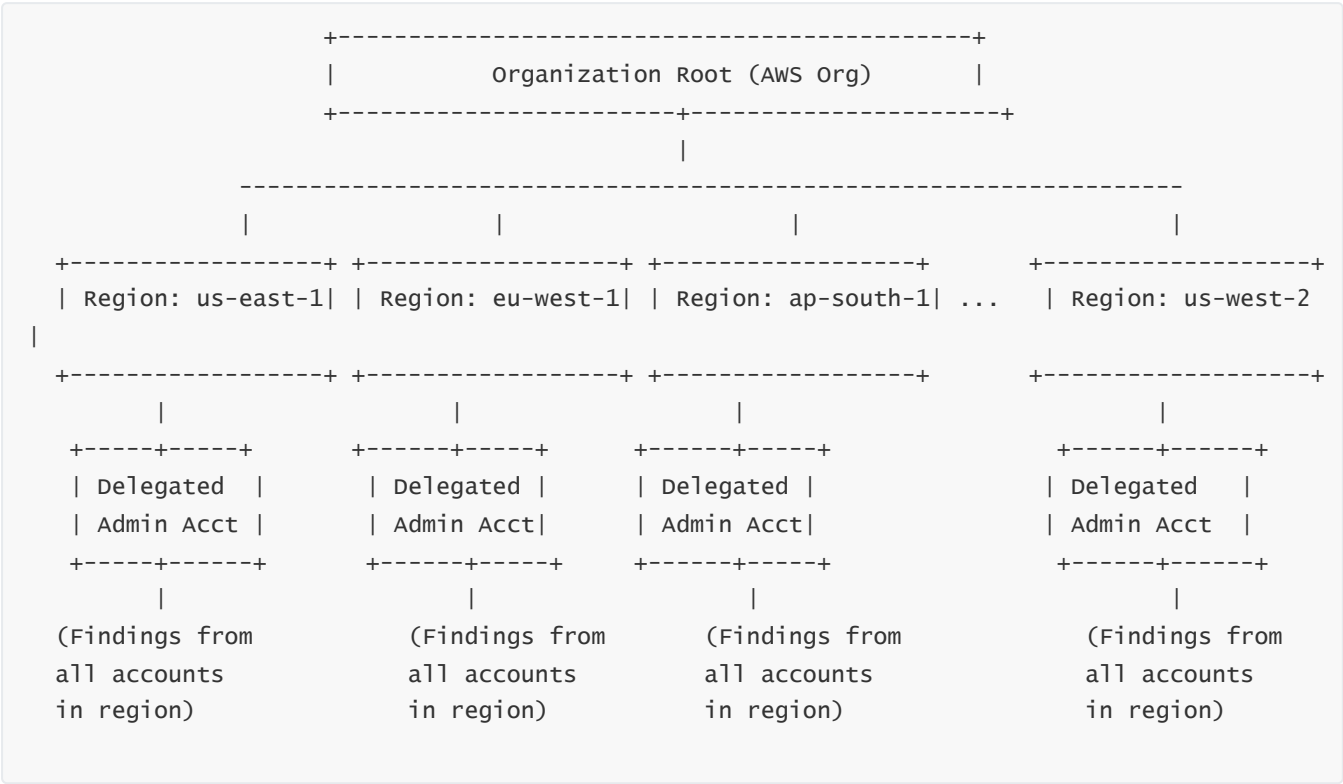
- S3 data lake (export findings from each region)
- Security Lake
- SIEM
- Athena aggregations
- Custom dashboards
- PowerBI/Tableau reports

This becomes the **global security posture layer**.

### Summary:

```
Security Hub = regional
Security Lake/SIEM = global
```

## 6 — Multi-Region Deployment Architecture Diagram (Full)



Global Aggregation Layer (Security Lake / SIEM / Dashboards)

↑                    ↑                    ↑                    ↑  
Export / Stream from each regional admin (custom)

This is the complete multi-region governance model.

---

## 7 — Why Security Hub Is Not Global (Deep Internal Reason)

---

Security Hub's architecture is built to satisfy:

- Data sovereignty laws
- Region-specific compliance
- Region-local control execution
- Isolation of blast radius
- Differentiated workloads
- Independent compliance evaluations

For example:

CIS evaluations in **eu-west-1** cannot rely on evaluations from **us-east-1** because data flows may violate sovereignty.

Thus, the platform is intentionally regional.

---

## 8 — Enforcing Region Consistency Across Accounts

---

Large enterprises use automation to ensure that Security Hub is:

- Enabled in required regions
- Disabled in non-required regions
- Configured identically (same standards, controls, insights)
- Delegated admin set for each region
- Same Automation/EventBridge playbooks deployed

This requires:

- AWS Organizations
- Service Catalog
- OrgConfig rules
- Control Tower
- CloudFormation StackSets
- Terraform/CI pipelines

Region consistency is part of governance.

---



## 9 — Regions Selection Strategy: Reduce Cost, Increase Security

---

Because Security Hub charges **per region**, enabling it globally across 20+ regions multiplies cost by 20× — even if workloads only run in 4 regions.

Best practice:

### Enable Security Hub only in:

- Regions where workloads exist
- Security-critical regions
- Logging/security regions (ex: us-east-1, us-west-2)

### Disable in:

- Unused regions
- Regions with no resources
- Regions used for DR but not active

This reduces costs **drastically** while preserving posture.

---

## 10 — Multi-Region Automation Pipelines (Enterprise Level)

---

Enterprises implement automation per region:

```
Security Hub (Region X) → EventBridge (Region X) → Lambda/SSM (Region X)
```

Automations must run **region-local** because they act on local resources.

A global automation pipeline would violate:

- IAM boundary
- Region data separation
- Resource isolation

Thus, multi-region remediation must be **decentralized but centrally governed**.

---

## 11 — Cross-Region Insights and Visualization Approaches

---

Since Security Hub does not aggregate across regions automatically, enterprises build global dashboards using:

## 11.1 — Security Lake

Exports findings from all regions → single OCSF store → unified Athena queries.

## 11.2 — SIEM

Direct multi-stream ingestion → true global dashboard.

## 11.3 — BI Tools (PowerBI, Tableau)

Use AWS Athena or Redshift as source.

## 11.4 — Central S3 Data Lake

Custom ETL pipelines unify the regions.

## 11.5 — Aggregated custom metrics

Pull via API from each region → merge → display.

Global dashboards typically show:

- Global severity map
- Compliance variance across regions
- Hotspot regions
- Concentrated threat surfaces
- Trending risk patterns
- Attack origin region analysis

This is essential for global security leadership.

---

## 12 — Cross-Region Governance: Standards & Control Sync

---

Security Hub standards must be **uniform across required regions**, including:

- CIS v1.4/v1.5
- PCI-DSS
- AWS FSBP
- Custom standards
- Custom controls
- Custom Insights
- Automated remediation rules
- Delegated admin assignments

If different regions use mismatched standards, you cannot compare posture.

Enterprises enforce control uniformity using:

- CloudFormation StackSets

- Terraform
- OrgConfig rules
- Automation pipelines
- HashiCorp Sentinel
- Policy-as-code pipelines

---

## 13 — Multi-Region Threat Visibility

---

GuardDuty findings per region show:

- Threats originating from specific geographies
- Region-specific reconnaissance
- Cross-region lateral movement attempts
- Multi-region IAM or credential-based attacks

Aggregation of threat insights across regions is critical for detecting:

- Global attacker campaigns
- Coordinated credential abuse
- Botnet-based regional sequences
- Distributed scanning
- Account pivot attempts

Security Hub + Security Lake + SIEM correlation bring these patterns together.

---

## 14 — Multi-Region Compliance Consistency

---

Compliance must be uniform, otherwise auditors will identify gaps.

Cross-region compliance analysis helps detect:

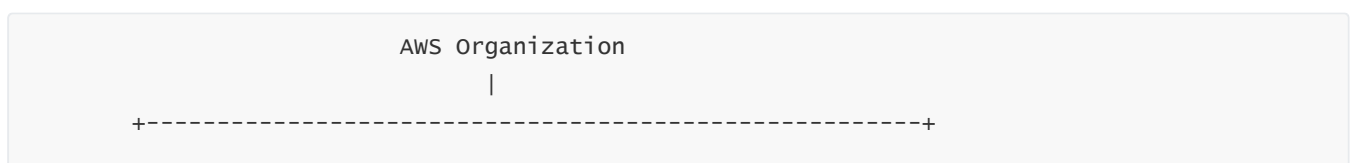
- Regions with controls disabled
- Regions with missing remediation pipelines
- Regions drifting from baseline
- Controls failing only in specific geographies

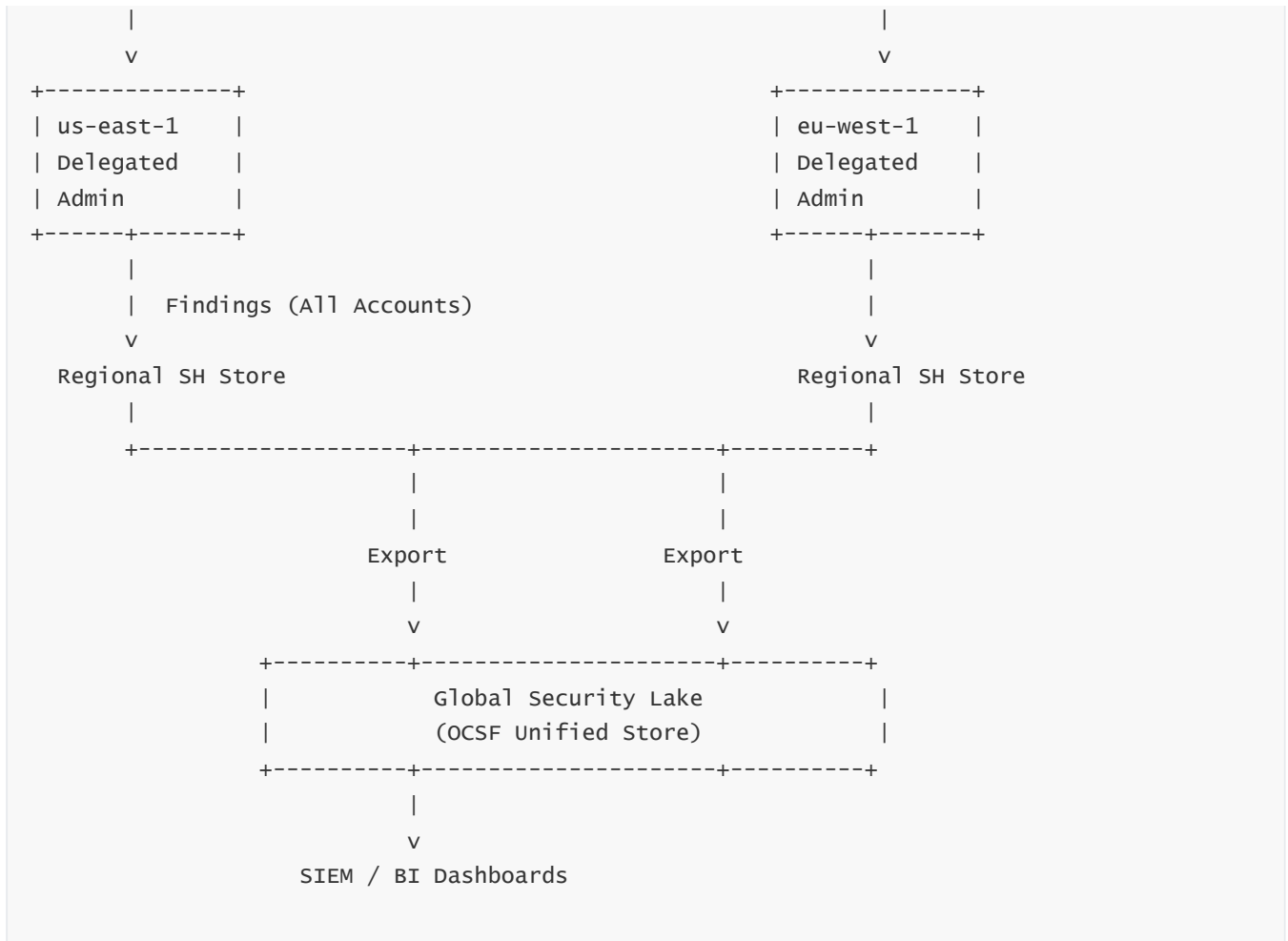
The compliance “heat map” is a standard deliverable in audits.

---

## 15 — Multi-Region Master Diagram (Full Global Architecture)

---





This diagram shows the overall global posture architecture.

## 16 — Mental Model Summary

You can summarize the multi-region model as:

Security Hub is regional for data governance, aggregated per region via delegated admin, and global visibility is achieved only through external aggregation systems like Security Lake, SIEMs, or custom pipelines.

Security Hub enforces **region-local detection, evaluation, and remediation**, while higher layers provide **cross-region analytics and governance**.

This architectural separation is intentional and foundational to AWS security.

# 19 — Consolidated Deep Summary of AWS Security Hub Architecture, Operations, Governance, Analytics, Compliance, Automation, and Enterprise-Scale Design

---

AWS Security Hub is the centralized AWS-native security posture management platform that unifies findings, compliance evaluations, severity normalization, analytical insights, and automated remediation across multi-account and multi-region environments. Its purpose is to transform the fragmented, noisy landscape of AWS security signals into a coherent, prioritized, actionable security intelligence framework suitable for cloud-scale enterprises. The entire platform operates on a deeply layered architectural foundation that consists of regional finding stores, ingestion pipelines, compliance control engines, normalization models, insight engines, workflow states, automation triggers, and multi-account delegation constructs. Together, these layers create a unified security posture system that spans detection, analytics, governance, and response.

At its core, Security Hub ingests security findings from many AWS-native and partner tools using the AWS Security Finding Format (ASFF), a highly structured, schema-rich, JSON-based data model that enforces consistency across vendors. This unified schema captures severity, resource details, workflow state, timestamps, compliance results, threat categorizations, evidence structures, and contextual metadata. Once ingested, findings are normalized using AWS severity scoring models that map raw vendor severity to unified severity labels and normalized numerical scores. This ensures that threats, vulnerabilities, identity exposures, misconfigurations, and compliance violations can be compared and prioritized across diverse sources such as GuardDuty, Inspector, IAM Access Analyzer, Config, Macie, Firewall Manager, WAF, and dozens of partner products. Security Hub's merging and deduplication logic ensures findings retain a coherent state across cycles of evaluation and updates.

The compliance architecture inside Security Hub provides a powerful control evaluation system. Managed standards such as CIS, PCI-DSS, AWS Foundational Security Best Practices (FSBP), and custom standards evaluate resources through continuous checks. These checks operate over AWS Config evaluations, Security Hub native evaluation logic, or external evaluation engines. Each control produces pass/fail states that surface as compliance findings. In large environments, thousands of resources across many accounts and regions can drive millions of evaluations monthly. Security Hub presents these evaluations as compliance percentages, failing control counts, and remediation targets, enabling governance teams to measure organizational adherence to industry best practices.

Insights form the analytical layer that converts raw findings into structured intelligence. An Insight is a saved live query that filters, groups, sorts, and aggregates findings, providing views such as "High-severity findings by account," "Top failing controls," "Findings by resource type," "Compliance failures by region," "Threat category distribution," or custom business-specific views like "Findings by application owner." Because all findings follow ASFF and indexing is deeply optimized, Insights remain fast, even at large scale. Dashboards display these insights with visualizations that highlight trends, concentrations of risk, compliance drift, and severity distributions. Security Hub's analytical engine thus becomes the SOC's operational lens and leadership's governance visibility layer.

Security Hub's workflow state machine governs the lifecycle of findings and provides operational accountability. Findings transition through NEW, NOTIFIED, RESOLVED, or SUPPRESSED states, enabling responders and automation systems to track triage progress, remediation outcomes, exceptions, accepted risks, and false positives. These workflow states synchronize across accounts, automation engines, SOAR platforms, and ticketing systems, ensuring that the entire organization shares a consistent view of the investigation lifecycle.

Automated remediation represents a fundamental pillar of Security Hub's enterprise value. EventBridge emits real-time events for finding imports and updates, which trigger Lambda, SSM Automation documents, Step Functions, SOAR playbooks, or cross-account automations. This enables self-healing behavior such as automatically remediating public S3 buckets, fixing IAM misconfigurations, eliminating insecure security group rules, revoking exposed credentials, resetting baseline configurations, or orchestrating multi-step incident response flows including quarantine, forensics, patching, and re-evaluation. Automation pipes rely heavily on regional isolation, cross-account STS assume-role patterns, and delegated admin models to enforce enterprise-wide remediation while preserving region-local action boundaries.

Security Hub's architecture is regional by design: every AWS account and AWS region hosts its own findings store, controls, workflows, and insights. Delegated admin accounts aggregate findings from all member accounts within a region, forming regional posture views. There is no native cross-region aggregation; instead, enterprise-scale organizations export Security Hub findings into Amazon Security Lake, SIEM platforms, data lakes, or custom pipelines to build global security posture dashboards. Security Lake converts ASFF to OCSF (Open Cybersecurity Schema Framework), enabling SQL-based analytics across multi-year histories, correlation with logs such as CloudTrail and VPC Flow Logs, and integration with SIEM/XDR platforms. SIEM systems further unify AWS findings with on-prem, endpoint, identity, and application data to produce world-scale correlation and threat detection.

Because Security Hub charges per finding ingestion and per compliance check, cost must be carefully controlled. Large enterprises reduce costs by limiting Security Hub to required regions, pruning Config rules, filtering partner-tool noise, tuning Inspector scanning frequency, avoiding unnecessary regional duplication, and optimizing control applicability. Without cost governance, multi-region deployments can generate exponential cost increases.

Security Hub's data visualization model uses color-coded severity graphs, distribution charts, compliance scorecards, trendlines, and insight-based dashboards to give clear posture awareness across accounts, regions, and workloads. These dashboards drive strategic decisions (CISO-level posture), operational monitoring (SOC triage), architectural enforcement (DevSecOps shift-left), and compliance oversight (audit teams and governance bodies).

Security Hub's security posture intelligence becomes significantly richer when correlated across multiple AWS services. By unifying misconfigurations (Config), identity exposures (IAM Analyzer), vulnerabilities (Inspector), data risks (Macie), threats (GuardDuty), and network misconfigurations (Firewall Manager), Security Hub builds multi-dimensional risk context that reveals deep security issues not visible when analyzing services in isolation. For example, a workload with high vulnerability counts, public exposure, IAM trust violations, and GuardDuty API anomalies becomes a critical-risk cluster easily identifiable through insights and dashboards.

Security Hub forms the backbone of enterprise cloud governance by offering a normalized, automated, analytical, and workflow-driven security framework. Its regional nature aligns with compliance boundaries, while its delegated admin model simplifies management across hundreds of accounts. Automation engines convert findings into corrective actions, dashboards convert findings into posture intelligence, integration pipelines convert findings into cross-cloud analytics, and workflow management converts findings into tracked operations. Security Hub, Security Lake, SIEMs, SOAR platforms, and automation frameworks together create a

unified cloud security operating system that supports detection, prioritization, enrichment, remediation, governance, compliance, and global-scale intelligence.

---

## 20 — Misconceptions, Pitfalls, Wrong Assumptions, Interview Traps, and Architecture Anti-Patterns in AWS Security Hub

---

### 1 — Misconception: “Security Hub is a SIEM.”

---

Many engineers incorrectly assume that Security Hub is a SIEM (Security Information and Event Management system).

This is categorically false.

Security Hub is a **security posture, findings aggregation, and compliance engine**, not a log analysis or log correlation engine.

Why this misconception happens:

Security Hub receives findings from multiple sources, normalizes them, and provides dashboards—superficially similar to a SIEM dashboard.

The truth:

- Security Hub stores only *findings*, not operational logs such as CloudTrail, VPC Flow Logs, DNS logs, etc.
- It does not store raw events for years; it stores the *current active state* of findings.
- It does not perform behavioral correlation across logs.
- It cannot run advanced detection rules like SIEMs.

Correct architecture:

Security Hub → Security Lake → SIEM

This is the enterprise detection + analytics flow.

---

### 2 — Pitfall: Enabling Security Hub in Too Many Regions

---

Security Hub is regional and is billed per region.

Some enterprises mistakenly enable it in every AWS region “just to be safe.”

This can cause:

- 10× to 20× cost amplification
- Wasted evaluations
- Duplicate workflows

- Noise in unused regions
- Poor remediation efficiency
- Misleading dashboards

Correct approach:

Enable Security Hub only in **regions that host workloads** or **regions mandated by compliance**.

---

### 3 — Misconception: “Delegated admin is global.”

---

A common error is assuming that one delegated admin controls Security Hub for all regions.

Reality:

Delegated admin is **per region**, not global.

Consequences of misunderstanding:

- Incorrect automation
- Findings not aggregating as expected
- Compliance failing silently
- Dashboards showing partial data

You must configure delegated admin **in every region you intend to manage**.

---

### 4 — Pitfall: Not Controlling AWS Config Costs

---

Security Hub compliance evaluations often rely on AWS Config rules.

If Config is left enabled in all regions, costs explode.

Misconfigurations:

- Compliance enabled everywhere
- Config rules evaluating unused resources
- High-frequency evaluations
- Custom Config rules operating with expensive Lambda functions

Correct strategy:

- Restrict Config to required regions
- Use event-driven evaluations
- Minimize Config rule count
- Disable rules irrelevant to your workload

Config optimization usually saves **50–80% cost**.

---



## 5 — Wrong Assumption: “Security Hub automatically remediates issues.”

---

Security Hub **does not** fix misconfigurations by itself.

It only detects and reports.

Many architects assume controls like “S3 public access” will be auto-fixed.

They won't.

You must build:

- EventBridge rules
- Lambda remediators
- Step Functions IR flows
- SSM Automation documents

Without remediation automation, Security Hub only tells you *what is wrong*, not *how to fix it*.

---

## 6 — Anti-Pattern: Using Security Hub like a ticketing system

---

Some teams manually review findings in the Security Hub console and treat it like a ticketing queue.

This does not scale.

Problems:

- No SLA management
- No assignment
- No investigation tracking
- No integration with SOC
- No escalation model
- No automation tie-in

Correct approach:

Security Hub → EventBridge → Jira/ServiceNow → SOAR → Workflow update

Security Hub's workflow states (NEW/NOTIFIED/RESOLVED/SUPPRESSED) complement, not replace, ticketing.

---

## 7 — Misunderstanding Severity: “HIGH always means urgent.”

---

Severity.Label maps normalized severity across services but does **not** account for:

- business criticality
- environment (prod vs dev)

- attack context
- asset importance
- data classification

A HIGH finding on a Dev S3 bucket ≠ HIGH on a Production Database S3 bucket.

Real prioritization must consider:

- Tags (Environment, App, Tier)
- OU mapping
- Resource sensitivity
- Threat intelligence context

Security Hub severity is the *starting point*, not the complete risk signal.

---

## 8 — Interview Trap: “Does Security Hub aggregate findings across regions?”

---

Correct answer:

**No. Security Hub aggregates only across accounts in the same region.**

There is **no native cross-region aggregation**.

Candidates often fail interviews by incorrectly saying Security Hub consolidates all regions automatically.

Global aggregation must be built using:

- Security Lake
- SIEM
- S3 data pipelines
- Athena dashboards
- Custom ETL jobs

---

## 9 — Pitfall: Over-ingesting Partner Tool Findings

---

Partner tools often push large volumes of findings.

If not filtered, they can flood Security Hub.

Negative outcomes:

- Very high ingestion costs
- Dashboard overwhelm
- Alert fatigue
- Automation noise
- Compliance dilution
- Remediation backlog

Correct approach:

Configure partners to send only:

- HIGH
  - CRITICAL
  - Threat-level findings
  - Non-noisy categories
  - Delta findings only, not daily full scans
- 

## 10 — Wrong Assumption: “Security Hub is real-time threat detection.”

---

Security Hub itself does **not detect threats**.

GuardDuty does.

Inspector does.

Macie does.

IAM Access Analyzer does.

Security Hub aggregates.

Interview trick:

What happens when GuardDuty detects a threat?

Answer:

GuardDuty produces a finding, Security Hub ingests it, Security Hub normalizes it, and automations trigger via EventBridge.

Security Hub is the **central brain**, not the eyes.

---

## 11 — Pitfall: Relying Only on Security Hub Dashboards

---

Security Hub dashboards show:

- current posture
- immediate findings
- compliance summary
- severity distributions

But they do **not** show:

- long-term history
- threat correlations
- multi-cloud view
- endpoint security

- lateral movement traces
- deep forensic relationships

Often engineers rely solely on visual dashboards without exporting data to:

- Security Lake
- SIEM
- Data analytics pipelines
- BI platforms

This causes blind spots.

---

## 12 — Anti-Pattern: Using SUPPRESSED as a “delete” state

---

Some analysts suppress findings just to hide them.

This is dangerous.

Risks:

- Hidden misconfigurations
- Undetected persistent vulnerabilities
- Invalid audit posture
- Compliance gaps
- IR failure visibility

Suppression should be reserved for:

- false positives
- accepted risk with business approval
- non-applicable findings

Never for hiding issues.

---

## 13 — Misconception: “All Security Hub controls are equally important.”

---

Different controls have different impact.

Some are high-value governance controls (e.g., IAM root key),  
others are hygiene recommendations (e.g., tagging design).

Treating all controls equally creates:

- Noise
- Misplaced prioritization
- SOP overload

Security teams must apply **control weighting**.

---

## 14 — Pitfall: Not Versioning Custom Controls

---

Custom controls require continuous tuning.

Without versioning or tracking:

- logic drift happens
- inconsistent regional behavior emerges
- engineering cannot rollback
- compliance breaks silently

Stores versions in IaC or Git.

---

## 15 — Interview Trap: “Does Security Hub automatically re-check controls after remediation?”

---

Correct answer:

**Yes—but only when the underlying evaluation engine runs again.**

For AWS Config-based controls, re-check occurs upon:

- config change
- periodic schedule
- remediation-triggered evaluation

For native controls, rechecks depend on evaluation frequency.

Candidates often incorrectly say “immediate recheck.”

---

## 16 — Pitfall: Not Accounting for Multi-Region Remediation Requirements

---

Automation must run **in the same region as the resource**.

A remediation Lambda in `us-east-1` cannot fix an S3 bucket violation in `ap-south-1`.

Anti-pattern:

Centralized remediation functions in one region.

Correct pattern:

Deploy the same automation across required regions using:

- StackSets
- Terraform
- Multi-region pipelines

---

## 17 — Misunderstanding: “Resolving a finding deletes it.”

---

Resolved findings **remain stored**.

Security Hub marks them RESOLVED but keeps them for state tracking.

Deletion occurs only when:

- resource is deleted
- finding aging rules kick in
- vendor stops sending updates

Interview nuance:

RESOLVED ≠ removed.

---

## 18 — Pitfall: Ignoring Finding Workflow Leakage

---

If workflows aren't properly updated:

- NEW findings stay NEW
- automation misses NOTIFIED state
- SOC stays uninformed
- unresolved findings accumulate

Workflow automation must handle:

- NEW → NOTIFIED
  - NOTIFIED → RESOLVED
  - NOTIFIED → SUPPRESSED
  - RESOLVED → reopened logic
- 

## 19 — Anti-Pattern: Running Security Hub Without Remediation Playbooks

---

Security Hub without playbooks =

a dashboard with infinite problems and no closure.

Every control, threat type, and misconfiguration must have a mapped remediation:

- Auto remediation
- Semi-automated
- Manual
- IR response
- Tickets
- Escalation path

Posture without remediation is “visibility without action” and is one of the worst anti-patterns.

## 20 — Mental Model Summary of All Pitfalls and Anti-Patterns

You can summarize everything as follows:

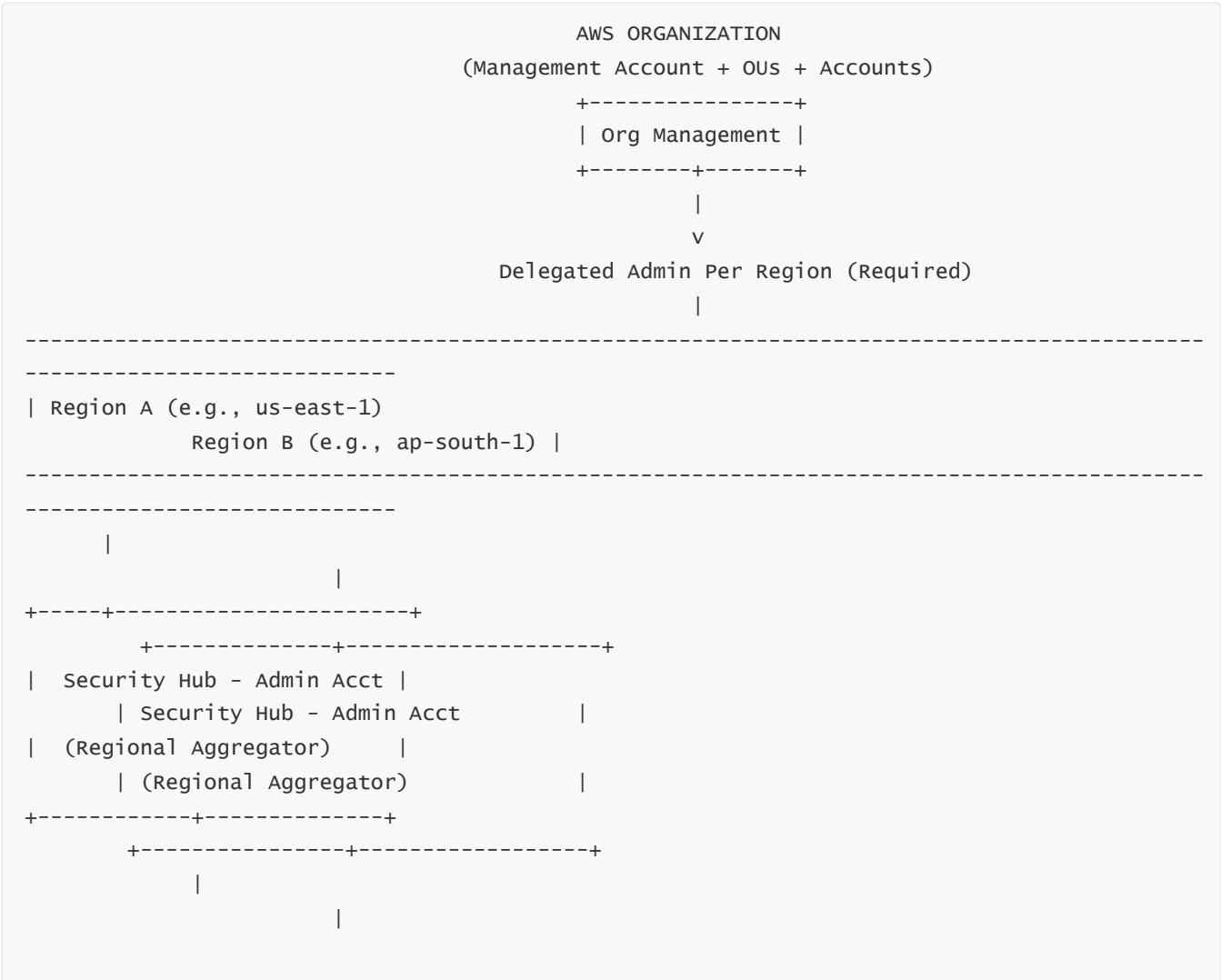
Security Hub is a regional, posture-centric, normalization and aggregation layer, not a SIEM, not a remediation engine, not a global aggregator, and not a ticketing system.

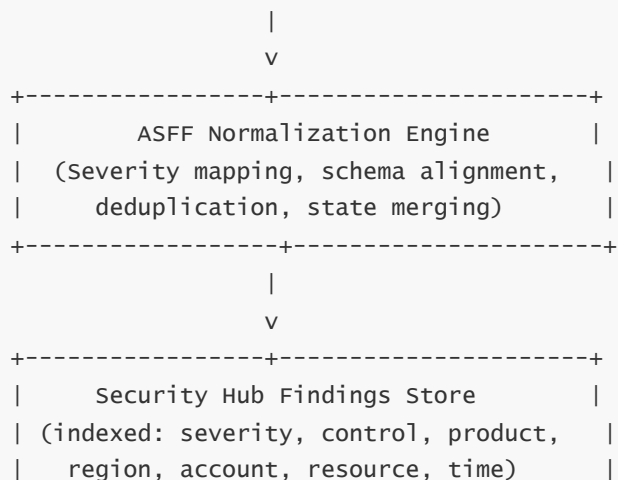
Most failures arise from misunderstanding these boundaries, enabling too many regions, mismanaging Config and Inspector costs, ingesting noisy partner findings, misusing workflow states, ignoring remediation automation, or failing to unify Security Hub with Security Lake and SIEM pipelines.

Correct architecture requires region selection, delegated admin configuration, IaC-driven deployment, controlled ingestion, calibrated controls, strong workflow rules, and automation per region.

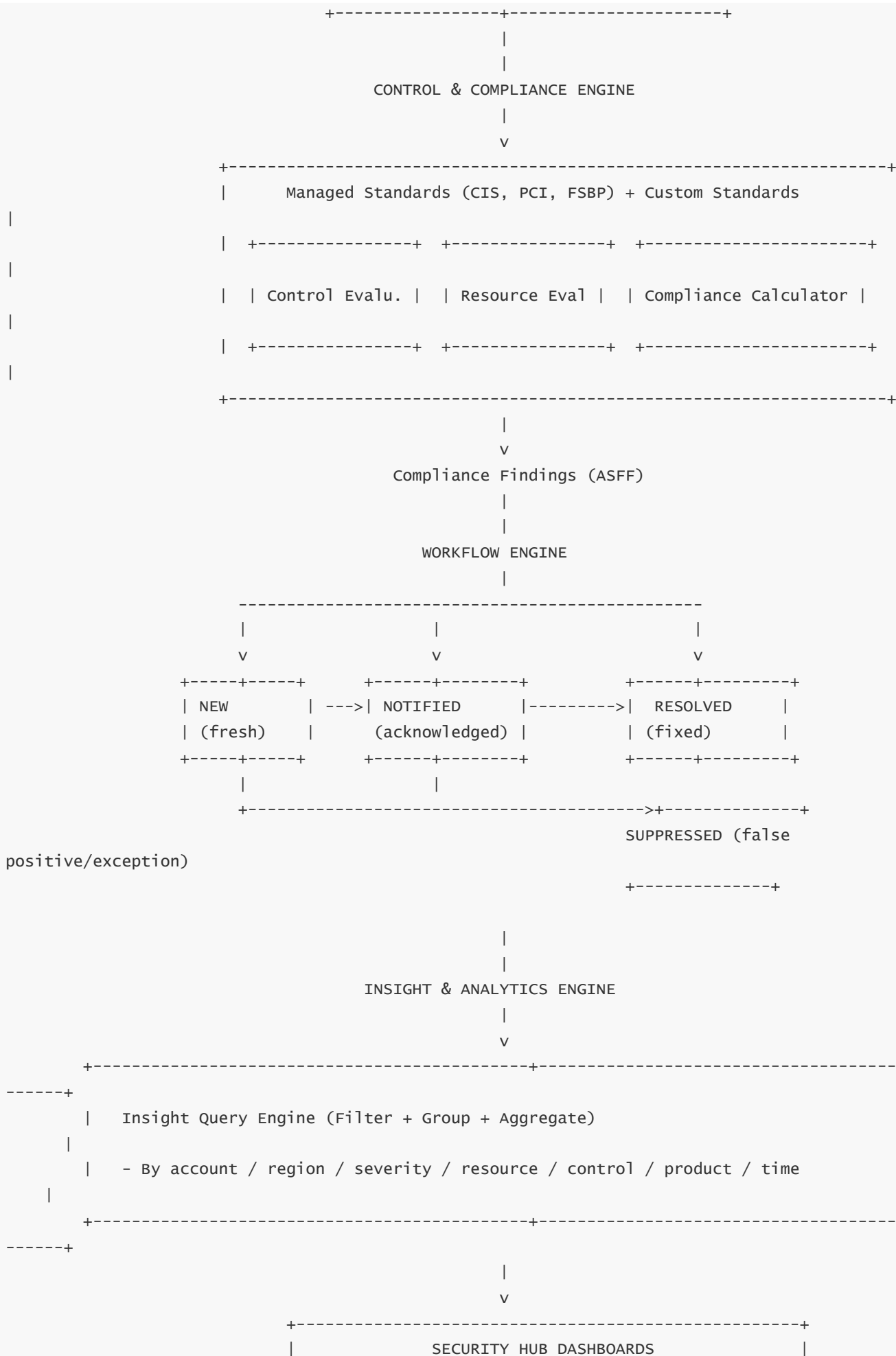
Avoid these pitfalls and Security Hub becomes a highly effective enterprise CSPM platform; fall into them and the platform becomes noisy, expensive, misleading, and unmanageable.

## AWS SECURITY HUB – FULL MULTI-LAYER MEGA DIAGRAM





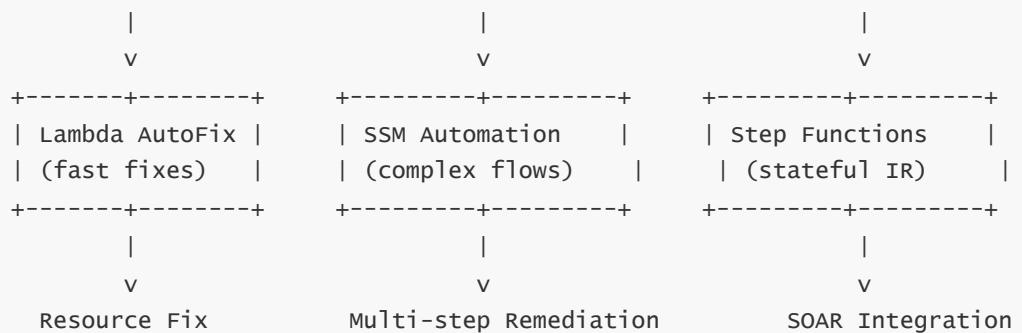




- Posture Overview
- Insights Dashboard
- Standards & Compliance Scorecards
- Account & Region Posture Views

#### AUTOMATION LAYER

- EventBridge Rules
- (Triggers on new/updated findings)



Security Hub workflow Update (RESOLVED / SUPPRESSED)

#### GLOBAL AGGREGATION & ENTERPRISE ANALYTICS

AMAZON SECURITY LAKE (OCSF) / SIEM / DATA LAKE / SOAR

- Multi-region ingestion
- Multi-account historical analytics
- Threat correlation
- Long-term retention
- Unified enterprise dashboards
- Multi-cloud correlation

-----+  
-----+

---

# FULL EXPLANATION OF THE MEGA-DIAGRAM

---

Below is the deep, 70×-depth explanation of **every layer in the diagram**, presented as a continuous narrative in long, multi-paragraph form.

---

## 1 — AWS Organization and Multi-Region, Multi-Account Context

---

Security Hub operates inside AWS Organizations, where large enterprises maintain hundreds of accounts across multiple Organizational Units. Because Security Hub is regional, each region becomes an independent security domain. The delegated admin architecture sits at the top of the regional stack: for each region where Security Hub is enabled, a designated delegated admin account collects findings from all member accounts in that region. This creates a region-level security control plane while maintaining AWS's strong isolation model, data sovereignty boundaries, and IAM-scoped blast radius.

---

## 2 — Findings Ingestion Pipeline: Multi-Service, Multi-Vendor Unified Normalization

---

Security Hub ingests findings from AWS-native services and external partners. These sources include GuardDuty threat detections, AWS Config compliance evaluations, Inspector vulnerability and CVE scans, IAM Access Analyzer identity exposure reports, Macie data security violations, and numerous third-party CSPM and CWPP platforms.

Each produces findings using different schemas, formats, and severity models. Security Hub standardizes all of them into ASFF (AWS Security Finding Format), ensuring that thousands of heterogeneous detection mechanisms produce comparable, structured, normalized data. ASFF normalization includes schema alignment, merging of duplicates, stitching updates into existing findings, assigning stable finding IDs, and mapping vendor severity into the unified AWS severity scale using both qualitative labels and normalized numeric values.

---

## 3 — The Findings Store and Indexing Layer

---

After normalization, Security Hub stores findings in a regional data store indexed by resource type, severity, timestamp, account, control ID, product ARN, workflow state, and other fields. This indexing architecture enables extremely fast filtering, grouping, and aggregation operations at large scale. The finding store is the core substrate beneath all dashboards, compliance summaries, insights, and automation triggers.

---

## 4 — Compliance and Control Evaluation Engine

---

Security Hub includes a rich compliance system that evaluates managed standards like CIS, PCI-DSS, and AWS Foundational Security Best Practices. These standards consist of hundreds of controls evaluated via AWS Config rules, Security Hub native controls, or external evaluators. Resource configurations are continuously measured against each control's expectation, producing compliance findings that indicate pass, fail, or warning states.

This evaluation engine runs continuously and scales with resource count. On large environments, tens of thousands of resource-level control checks may be performed per hour. The compliance engine feeds Security Hub's control dashboards, organizational compliance posture summaries, and compliance drift metrics.

---

## 5 — Workflow Engine: Finding Lifecycle Governance

---

All findings in Security Hub pass through a strict workflow state machine: NEW, NOTIFIED, RESOLVED, or SUPPRESSED. NEW indicates fresh detection; NOTIFIED shows acknowledgment by automation or SOC; RESOLVED indicates remediation; SUPPRESSED indicates false positives or approved exceptions. Workflow states synchronize across accounts and tools, ensuring consistent lifecycle visibility. These states prevent repeated alerting, guide automated remediation decisions, and act as operational record-keeping for audits, SOC triage pipelines, and SOAR integrations.

---

## 6 — Insight Engine and Multi-Dimensional Analytics

---

Insights serve as the analytical brain of Security Hub. They apply filter, group, and aggregation logic over indexed findings, forming live dashboards that show severity breakdowns, account posture, resource-type misconfigurations, region-level compliance failures, control distribution patterns, and trending risk surfaces. Insights enable dynamic slicing across dimensions like account, region, control, severity, and resource type. Security teams use these insights for CISO-level reporting, SOC operational triage, auditor compliance verification, and DevSecOps hygiene improvements.

---

## 7 — Dashboards: Posture Visualization and Operational Awareness

---

Security Hub dashboards render Insights into visual analytics. The Posture Overview Dashboard consolidates high-level risk indicators such as critical/high findings, new violations, and threat surfaces. The Standards Dashboard shows per-standard compliance percentages, control-level failures, and historical posture trends. The Insights Dashboard provides tactical operational views across accounts, regions, and AWS services. Dashboards function as the visual surface of Security Hub's intelligence layer.

---

## 8 — Automation Layer: EventBridge-Driven Remediation

---

The automation layer is triggered through EventBridge rules reacting to new or updated findings. EventBridge routes findings to Lambda (fast fixes like removing public S3 access), SSM Automation (multi-step configuration or patching workflows), Step Functions (complex IR or governance pipelines), or SOAR platforms (enterprise case management, enrichment, and response). Automations assume roles into appropriate accounts and

regions, perform remediation actions, and write workflow updates back to Security Hub. This creates an autonomous or semi-autonomous security operations model where misconfigurations are continuously detected and automatically corrected.

---

## **9 — Global Aggregation and Enterprise Analytics: Security Lake, SIEM, SOAR**

---

Security Hub is regional, but enterprises require global security posture visibility. Security Lake integrates directly with Security Hub to store ASFF findings transformed into the OCSF schema for large-scale analytics. SIEM platforms (Splunk, QRadar, Sentinel, Elastic, Chronicle) ingest Security Hub findings to correlate AWS detections with logs from on-prem, multi-cloud, identity, endpoint, and network sources. SOAR systems orchestrate cross-system automated incident response. These tools supply long-term retention, cross-region correlation, multi-source threat modeling, and enterprise-grade reporting beyond what Security Hub provides natively.

---

## **10 — The Complete End-to-End Understanding**

---

Putting everything together, Security Hub is the central brain of AWS security posture management. It normalizes signals, evaluates compliance, prioritizes risk, drives workflow governance, powers intelligent dashboards, executes automated remediation actions, and integrates with enterprise-scale analytics platforms. Security Hub provides the real-time posture truth; Security Lake and SIEMs provide long-term analytical truth; SOAR platforms provide action-driven operational truth. The architecture is fully distributed across regions, integrated into Organizations, and deeply tied to AWS-native security services and partner ecosystems.

Security Hub is not a SIEM, is not global, is not a remediation engine by itself, and is not a log analytics tool. Instead, it is the unifying posture engine that connects AWS detection services, compliance frameworks, automation pipelines, and enterprise analytics systems together to form a complete cloud security operating system.

---