Best Practices for Security Data Pipelines

Getting Security Telemetry from Source to Storage and Ready to Use

Table of Contents

Why Security Teams Move Data 1

What Approach Should I Take in Planning My Security Data Pipeline Project? 3

What Technical Topics Should I Be Concerned With in Planning My Pipelines? 4

How Do I Shape Security Data for Use in Analytics, Threat Hunting, Detection Engineering, and Investigations? 5

What Best Practices Should I Consider? 6

Why Does the Format and Partitioning of Security Data Matter? 7

What Is the Typical ROI of a Security Data Pipeline Project When Done Right? 8

How Query Can Help 9

Why Security Teams Move Data

Security teams are dealing with more telemetry than ever before from EDR products, identity providers, SaaS apps, network tools, cloud infrastructure, and more. Every alert, login, configuration change, and packet capture becomes part of a growing data exhaust that is relevant for security operations work.

Keeping all of that data in a SIEM isn't realistic for most organizations. Ingest-based pricing models make centralizing everything cost-prohibitive, and even "search credit" billing quickly becomes unpredictable as data volumes grow. The math just doesn't work.

Source tools aren't a reliable archive either. Many only store logs for 30 days before they roll off, leaving you blind to older activity or long-tail threats. That's a problem when investigations routinely stretch back months, and compliance frameworks demand 12 months or more of retained data.

The result is that security teams need a way to offload telemetry from expensive, short-term systems into a long-term, cost-efficient repository without losing the ability to search and analyze it later. **At a certain point, the question isn't whether you need to keep the data. It's where you keep your data.**

For most modern security programs, that repository is often public cloud object storage. Platforms like Amazon S3, Azure Storage Account Blob Containers (e.g., ADLSv2), and Google Cloud Storage offer the scalability and economics to keep more data for longer, while giving teams the control to shape and query it on their own terms.

This whitepaper focuses on sharing key considerations and best practices for building pipelines that move security telemetry from its source to cloud storage in a format that is clean, structured, and immediately ready for use.

Security teams move data to cloud storage so they can:

- **Decouple data storage from search and alerting tools**
  Rather than being locked into a SIEM's ingest-based pricing model, teams use cloud storage to offload raw or enriched telemetry and reduce costs.
- **Retain more data, longer**
  Many teams need 12–24 months of logs for compliance or investigations. With SIEMs as your hot storage, that's unaffordable. Cloud object storage makes it possible.
- **Shape data for modern detection and analytics use cases**
  Public cloud storage becomes the foundation of a security data lake, enabling teams to run detections, hunt threats, or power investigations using scalable query engines, without rehydrating cold storage.
- **Enable vendor-agnostic architectures**
  By writing data once to a clean, ready-to-use format like Parquet, teams can plug in tools like Trino, BigQuery, or Spark; or feed the data into vector databases and AI pipelines. No lock-in.
- **Prepare for AI, not just SIEM**
  Whether it's powering retrieval-augmented generation (RAG), enriching alerts, or training models, AI-ready security data starts with owning your own logs in a structured, accessible format.

Cloud object storage is becoming the default destination for security telemetry. But moving data to the cloud is only part of the story.

What matters most is **how it lands.** It should be normalized, partitioned, and structured in a way that makes it immediately useful for detection, analytics, and investigations.

What Approach Should I Take in Planning My Security Data Pipeline Project?

Most security data pipeline projects start with a simple goal: just move the data. But the good ones start with a better question: What do we actually need this data to do?

Before you start wiring up connectors or writing jobs, you need a plan. That means thinking beyond raw ingestion to what the pipeline needs to support across security operations, engineering, and compliance.

We recommend thinking in three layers:

Downstream consumers

Think SOC analysts, detection engineers, threat hunters, incident responders, GRC teams, and even AI workflows. What are they using the data for? What do they need from this data? How will they access it?

The technical enablers like partitioning, compression, and normalized schema matter, but only in service of these outcomes. Start with the use cases, then work backward to what the data needs to look like.

Source coverage

Identify which sources you need to pull from: EDRs like CrowdStrike, identity providers like Entra ID or Okta, network security tools, SaaS apps, and more.

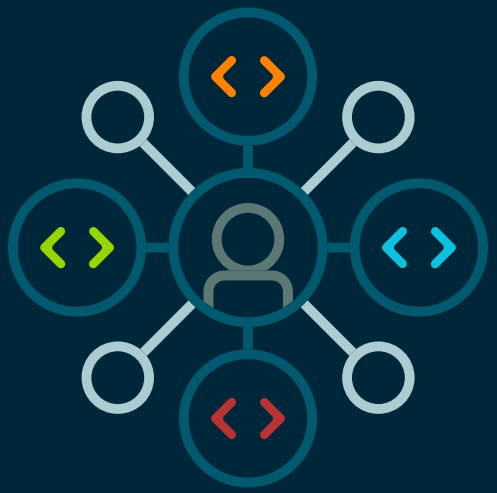
Don't just list them. Classify them by type, priority, and use case: high-fidelity detection sources, enrichment-only sources, long-tail compliance requirements, etc.

Destination and format strategy

Are you writing to Amazon S3, Azure Blob, Google Cloud Storage, or all three? Do you need Parquet? OCSF? Delta Lake? Planning the structure and format of your security data upfront is critical for enabling downstream efficiency.

A well-architected security data pipeline doesn't just move logs, it delivers ready-to-use data to the right place, in the right shape, with the right controls.

Remember: without planning, you risk building brittle, one-off jobs that don't scale. Or worse: you land unstructured, bloated telemetry that no one can use.



What Technical Topics Should I Be Concerned With in Planning My Pipelines?

Once you know who needs the data and what they're trying to do with it, the next step is making sure the pipeline actually delivers. That means getting a handful of technical decisions right, the kind that quietly make or break performance, cost, and maintainability down the line.

Here are some key areas to focus on:

Data shaping and normalization

Raw telemetry is messy. Standardizing on a schema, like OCSF, isn't always required, but it can deliver significant benefits. A consistent structure makes it easier to join data across tools, write detections once, and ensure compatibility with search or analytics engines.

Cleaning up noisy fields and dropping irrelevant ones also reduces storage cost and improves usability. Without some level of shaping, every downstream user ends up reinventing the normalization wheel.

Compression and file format

Writing security data in Parquet and compressing it with something like Zstandard (ZSTD) or Snappy can shrink your storage footprint by up to 80% compared to JSON. This also makes it dramatically cheaper to scan and query later.

Partitioning for performance

Partitioning your data (e.g., by source, event type, timestamp) is essential for large-scale search and analytics. Whether you're using Athena, BigQuery, or a federated search solution, smart partitioning means faster queries and lower costs.

Historical hydration

Sometimes you need to backfill days or weeks of data when first setting up a pipeline. Make sure your system supports historical pulls, ideally in a way that doesn't flood your storage or break your budget.

Scheduling and recurrence

Decide how often data should move. Every minute? Hour? Day? The right cadence depends on your use case. Real-time detections require faster flows than quarterly audit logs.

Access controls and governance

You're not just moving data, you're moving sensitive data. Tag it, track it, and restrict access to only what's needed. The more usable your pipeline becomes, the more teams will want in.

Think of these not as nice-to-haves, but as foundational design decisions. When they're baked into your pipeline from the beginning, everything downstream, from threat hunting to AI, gets easier.

How Do I Shape Security Data for Use in Analytics, Threat Hunting, Detection Engineering, and Investigations?

Security data becomes valuable the moment it can be searched, filtered, and joined with other data. But that value only materializes if the data is shaped correctly. You're aiming for clean enough to trust, structured enough to query, and performant enough to support real-time workflows.

Here's what that looks like in practice:

Flatten complex nested structures

Most security tools output JSON full of nested fields, arrays, and blobs. That might work for a single product's UI, but it breaks down when trying to correlate across tools or scan with a query engine.

Flattening or normalizing fields - especially for assets, indicators, and timestamps - makes the data more usable across detection and investigation workflows.

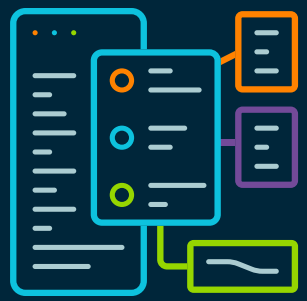
Additionally, popular tools such as Polars and Arrow require explicit schemas to be defined when dealing with complex nested data, as well as high performance data warehouse platforms.

Standardize event types and timestamps

It's hard to run analytics or threat hunts when each log source uses a different time format or event label. Whether you use OCSF or a custom taxonomy, enforcing consistency helps you ask questions like "Show me all failed authentications from internal IPs" across multiple vendors without rewriting every query.

Pre-filter and pre-shape where possible

Not every field from every event is worth keeping. By shaping the data during pipeline execution - e.g., dropping verbose or duplicative fields, excluding unactionable events - you reduce storage overhead and focus downstream effort on the data that matters.



Partition for the workflows, not just the storage

Don't just think in terms of "write performance", think in terms of "read efficiency." Threat hunters often pivot by user, IP, or timeframe. Partitioning and organizing data to match these patterns makes investigations significantly faster. Go further by expanding to clusters, Z-Ordering, and/or using Bloom Filters if your data engines support it.

Choose formats and structures that scale

Columnar binary file formats like Parquet, combined with compression (ZStandard, Snappy), enable cheaper storage while maintaining fast query speeds for codecs that can be decompressed quickly. But just as important is how the data is organized within those files; consistent column names, minimal nested objects, and consistent overall schema.

When you shape security data to be ready-to-use, you cut down on triage time, reduce toil for end users, and make every investigation faster. Good shape means faster answers.

What Best Practices Should I Consider?

If you're building or reworking a security data pipeline, best practices aren't just nice to have, they're how you avoid brittle systems, blown budgets, and unusable data.

Based on what we've seen across dozens of teams, here's what actually works:

Start with the use case, not the tool

Too many teams start with a connector or platform in mind, then retrofit their use case to whatever comes out the other end. Flip that. Work backward from what your team needs to detect, investigate, or analyze.

Filter early, not late

The earlier you reduce noise at the source or within the pipeline, the less data you have to store, move, and process. Pre-filtering logs you don't need for detection or investigation helps you avoid paying a tax on telemetry that just takes up space.

Write once, read many

Avoid building single-purpose flows. Instead, write clean, structured data into cloud storage in a format that supports multiple consumers: search tools, analytics engines, data science workflows, and even SIEMs.

Optimize for read performance, not just write simplicity

What looks easy on the write side often becomes painful on the read side. Partitioning, compression, and schema consistency all make life better for downstream users, and cheaper for your team.

Keep infrastructure lightweight

Heavy-handed ETL platforms and homegrown data plumbing often collapse under their own weight. Look for approaches that minimize moving parts: simple scheduling, stateless jobs, and native cloud storage integrations.

Plan for governance up front

If the pipeline works, more teams will want access. Build with RBAC, tagging, and access policies from the start so you're not untangling permissions later.

Most importantly: don't treat this as a one-and-done project. The best teams treat their pipelines like any other production system by constantly measuring performance and continuously improving the process based on feedback from consumers and producers.

Why Does the Format and Partitioning of Security Data Matter?

The format and partitioning of your data aren't just implementation details, they directly impact how fast your team can ask questions, how much it costs to store and analyze data, and how usable that data is across workflows.

Here's why they matter:

Query performance

Choosing a columnar format like Parquet enables engines like Athena, BigQuery, and Spark to scan only what's needed.

Whether you are picking columns out of a DataFrame, using an exploratory data analysis engine such as Azure Data Explorer or DuckDB, or executing queries with Clickhouse, Snowflake, or Databricks, columnar formats partnered with querying the data with your exact columns will vastly speed up query plans and execution and lower overall scan sizes.

Cost efficiency

JSON is easy to generate but expensive to store and process. Formats like Parquet can reduce file sizes by 70-80%, especially when paired with compression. Partitioning further helps by limiting how much data needs to be scanned on each query if your upstream engine supports pruning datasets based on those partitions.

Going further and clustering your partitions, applying secondary clustering to tables (such as in Google BigQuery or Databricks), and using Bloom Filters can further optimize your costs and query performance.

Workflow alignment

Partitioning your data by time, source, and event type makes it easier for downstream tools and analysts to find what they need.

Without it, even basic triage queries can become painfully slow due to full table scans. Again, ensure your query engine or data engine of choice supports pruning by partitions.

Scalability and maintenance

Well-partitioned data in the right format scales better with increasing data volumes. Instead of reprocessing full datasets, modern engines can target just the relevant slices.

Likewise, maintenance such as compaction jobs to reduce the "small file problem" become much more simplistic with standardized partitioning.

AI and automation readiness

Structured, partitioned data makes it easier to power ML pipelines, feed vector databases, or enrich alerts using real-time context. Clean inputs yield better AI outputs.

In summary: format and partitioning aren't just for your data engineers. They're foundational to the performance, cost, and usability of your entire security data stack.

What Is the Typical ROI of a Security Data Pipeline Project When Done Right?

Security data pipeline projects don't always come with a neat ROI calculator, but when done right, the returns are clear, compounding, and measurable across several dimensions.

Here's where teams typically see the payoff:

SIEM cost reduction

By filtering and routing raw telemetry directly to cloud storage, teams often reduce SIEM ingest volume by 30–70%.[1] That alone can mean hundreds of thousands in annual savings for large orgs.

Faster investigation and response times

With well-partitioned, structured data in a queryable format, analysts spend less time pulling context and more time making decisions. Threat hunting cycles shrink from hours to minutes.

Lower storage and compute costs

Writing data in Parquet and compressing it with ZSTD or Snappy slashes storage bills and compute spend for downstream analytics.[2] You keep more data, for longer, at a fraction of the cost.

Less engineering maintenance

When pipelines are simple, stateless, and infrastructure-light, you won't need three engineers babysitting brittle ETL code that breaks every time a timestamp format changes.

Increased data usability across teams

Structured, well-governed data doesn't just help the SOC, it helps compliance, GRC, threat intel, and AI teams work from the same source of truth.

Enablement of new workflows

Clean, ready-to-use data unlocks use cases that were previously cost-prohibitive: broader detection coverage, long-term trend analysis, or enriching LLMs with real-time context.

And maybe most importantly: these aren't tradeoffs. A well-designed pipeline gives you more control, more data, and better performance, at lower cost.[3]

How Query Can Help

Query has launched its Security Data Pipelines solution. It's built for teams that want a simpler way to move security telemetry to cloud storage, without building brittle ETL jobs or buying heavyweight infrastructure.

With Query, you can:

- Ingest from tools like CrowdStrike, Entra ID, and dozens more
- Land telemetry in Parquet format, compressed and partitioned
- Control what data moves, how often, and how far back to go
- Skip the hassle of writing transformations, normalizers, or schedulers

It's everything you need to write to the gold layer of your security lake, clean, structured, and ready to use.

Want to see it in action?

Learn more about Query Security Data Pipelines

[1] *Priority Logs for SIEM Ingestion – GitHub*
[2] *IJSAT Volume 4 Issue 3, 2024*
[3] *Apache Parquet Whitepaper – arXiv 2304.05028*