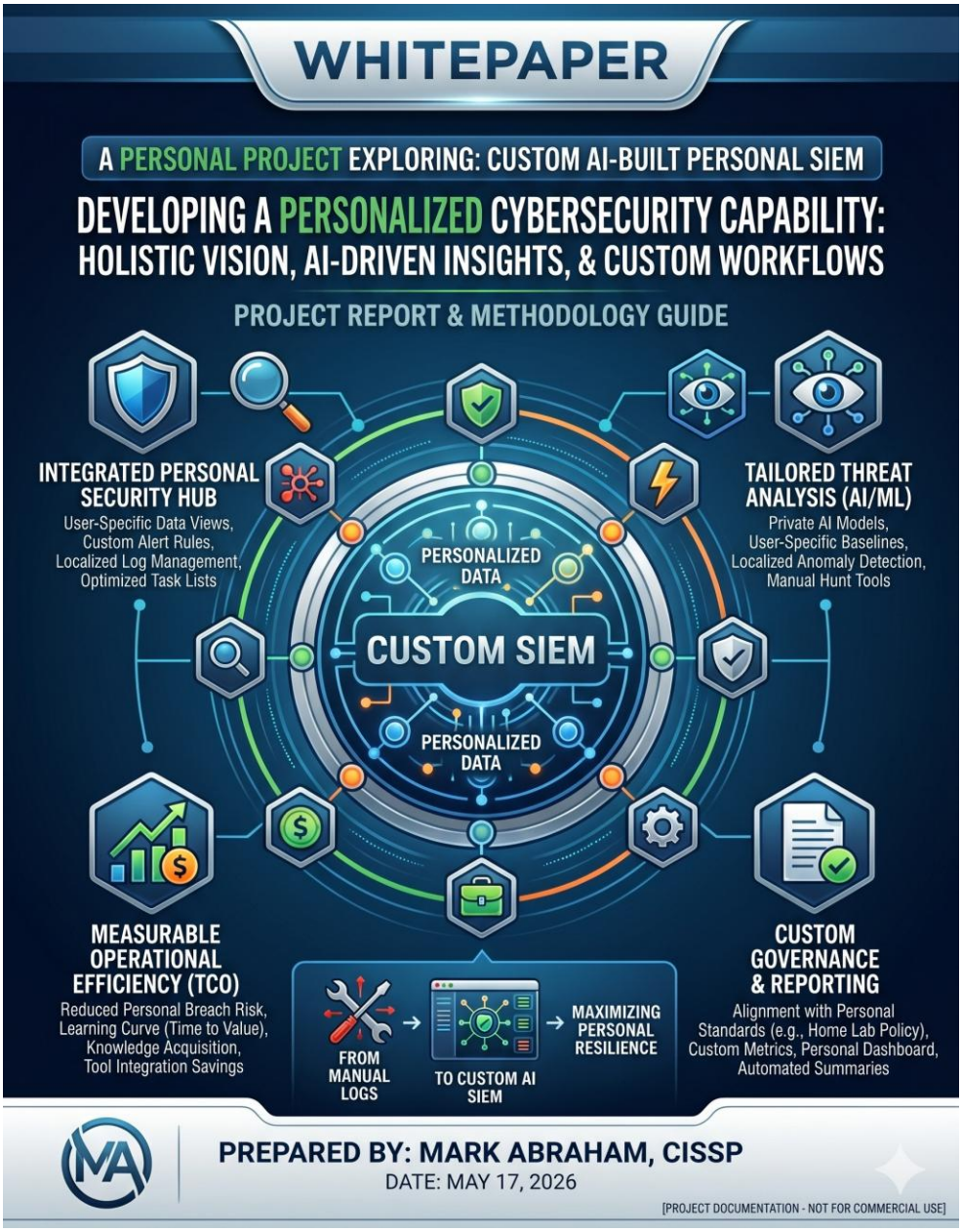
ZillaSIEM™

AI-Assisted Local SIEM & Network Defense Platform

Telemetry • Enrichment • Detection • Analyst Workflows • Operational Validation

This whitepaper documents the architecture, telemetry pipelines, operational workflows, detection engineering, and AI-assisted engineering processes used to build the ZillaSIEM™ local SIEM/NDR platform.



Foreword

This white paper was never supposed to exist.

ZillaSIEM™ did not begin as a commercial product, a startup idea, or a compliance exercise. It started as curiosity — specifically, a deep desire to understand local AI systems at a much more practical and operational level.

Like many people watching the rapid acceleration of AI capabilities, I found myself asking a simple question:

What happens when highly capable AI systems become accessible to everyone?

Not just researchers.
Not just governments.
Not just large enterprises.

Everyone.

That question matters because the reality is uncomfortable: AI is going to make many things easier — including offensive capability development, reconnaissance, exploitation research, social engineering, automation of attack workflows, and scalable operational targeting. Capabilities that once required highly specialized expertise are rapidly becoming more accessible, more automated, and far easier to operationalize.

Security professionals everywhere are trying to grapple with what that means.

Some people are responding with panic.
Some with denial.
Some with blind optimism.

My personal belief is much simpler:

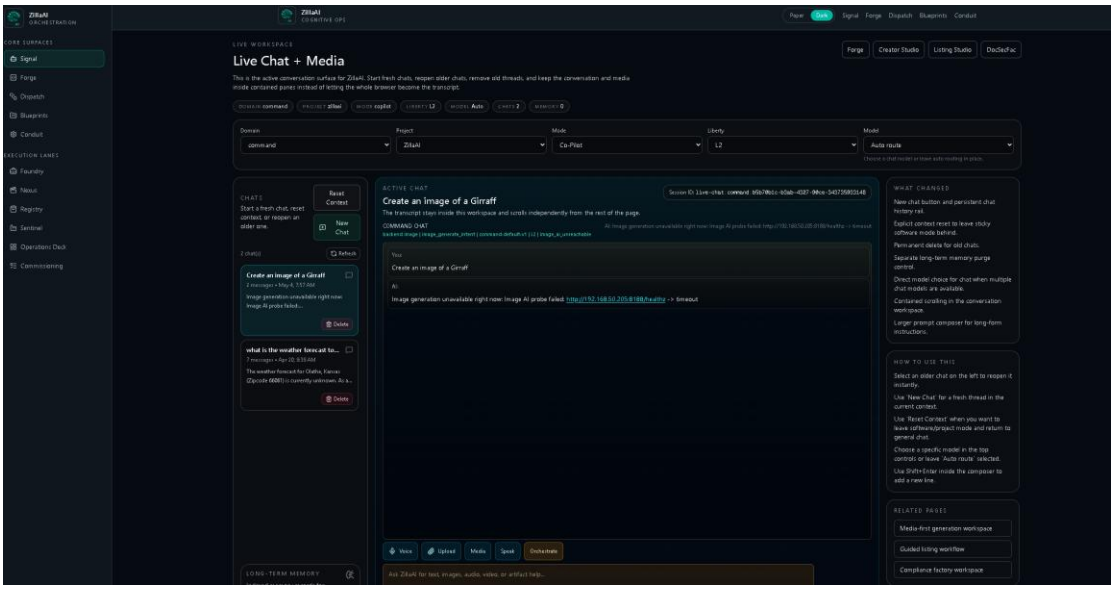
"You can fear AI, or you can learn it."

The models are not going away.
The tooling is not slowing down.
The barrier to entry is collapsing in real time.

That means defenders have a choice to make. Either we continue approaching security using workflows designed for a pre-AI world, or we adapt and become dramatically better at telemetry, operational awareness, evidence correlation, IOC hunting, and visibility into increasingly intelligent systems.

Ironically, what eventually became ZillaSIEM™ actually started somewhere else entirely:

Enter **ZillaAI™!**



ZillaAI was — and still is — the original project.

The original vision was never simply “build a chatbot.” The goal was to build a deeply capable local AI ecosystem centered around orchestration, modular tooling, operational context, workflow coordination, and intelligent assistant behavior that felt substantially more useful, extensible, and operationally aware than many of the open-source assistant frameworks emerging at the time.

As the orchestration capabilities expanded, something became painfully obvious:

If an AI system becomes deeply integrated into your operational life, workflows, infrastructure, and decision-making processes… it becomes one of the highest-value targets you own.

That realization triggered what I can only describe as a full security “spider-sense” moment.

Red alert!

The deeper the orchestration became, the more obvious it was that traditional consumer-grade visibility and monitoring approaches were nowhere near sufficient for the level of operational awareness I wanted surrounding a system like ZillaAI.

I did not just need logs.

I needed:

- telemetry
- correlation

- enrichment
- runtime visibility
- infrastructure awareness
- threat hunting
- evidence retention
- operational monitoring
- AI-assisted analysis
- continuous validation

In many ways:

“ZillaSIEM™ was born because ZillaAI needed protection.”

The desire to build a smarter AI ecosystem directly created the need for a smarter security and observability ecosystem around it.

One of the more unusual aspects of this project is that while I served as the enterprise development architect and security architect for the environment, I did not personally write any of the implementation code powering the ecosystem.

Instead, I directed AI systems — primarily Codex-driven engineering workflows — to build large portions of the operational environment under continuous human governance, architectural steering, correction, validation, and security oversight.

“That distinction matters.”

This was not simple autocomplete assistance or isolated code generation.

The workflow evolved into something much closer to:

human-governed AI systems engineering

The AI performed most of the implementation labor:

- coding
- refactoring
- integration work
- service generation
- troubleshooting support

- deployment adjustments
- configuration generation
- operational scripting
- documentation support

My role increasingly became:

- architecture
- orchestration
- governance
- security direction
- operational constraint design
- systems thinking
- validation
- strategic steering
- discipline enforcement

In many cases, the most valuable contribution was not writing code, but teaching the AI to think differently — more operationally, more defensively, more systematically, and more cautiously about infrastructure, telemetry, maintainability, trust boundaries, and long-term operational behavior.

Over time, the environment expanded beyond AI-assisted coding and evolved into something much larger:

an orchestrated AI engineering ecosystem

Rather than using generic assistants, the ecosystem incorporated specialized expert-oriented personas designed around different operational disciplines and reasoning perspectives.

These included:

- UX and usability specialists
- security architects
- enterprise architecture reviewers

- operational analysts
- QA and testing personnel
- refinement specialists
- workflow coordinators
- governance-oriented reviewers
- systems-thinking evaluators

The environment eventually grew into a roughly fifty-persona AI operational staff designed to simulate portions of a functional engineering and operational organization.

Importantly, these personas were not limited to static prompting behavior.

They could:

- conduct targeted research *"Literally use a browser to go hunt answers!"*
- retrieve supporting information
- critique implementation approaches
- challenge assumptions
- participate in structured reviews
- evaluate workflows
- provide domain-specific reasoning perspectives

The result was an environment where AI systems increasingly behaved less like isolated tools and more like a coordinated operational staff operating under human governance and architectural oversight.

"That produced surprisingly valuable outcomes."

Security-oriented personas would identify trust-boundary concerns.
UX-focused personas would challenge dashboard clarity and workflow friction.
Architecture-focused personas would critique scalability and modularity assumptions.
Operational reviewers would question maintainability, telemetry continuity, and evidence-retention strategy.

The interaction between these perspectives regularly surfaced weaknesses and blind spots that likely would have gone unnoticed in a traditional single-perspective development workflow.

The orchestration environment also evolved beyond static development assistance.

I trained ZillaAI to:

- coordinate operational workflows
- direct work to Codex
- maintain implementation continuity
- sequence refinement tasks
- orchestrate validation activities
- manage ongoing engineering momentum

Playwright-driven testing workflows were integrated to support:

- smoke testing
- WebGUI validation
- interface testing
- workflow verification
- operational review automation

This meant that while I was at work or otherwise occupied, the ecosystem itself could continue progressing implementation tasks, refinement cycles, validation workflows, testing activities, and operational iteration within bounded constraints.

That experience fundamentally changed my understanding of where software engineering is heading.

"I no longer believe the highest-value engineering role will necessarily belong to the person typing the most code."

Increasingly, the highest leverage may belong to the people who can:

- architect systems
- govern AI behavior
- validate outcomes
- maintain discipline
- identify risks
- orchestrate workflows
- think operationally

- steer intelligent implementation ecosystems effectively

In many ways, ZillaSIEM™ itself became both:

- a security platform
- and a live experiment in AI-directed infrastructure engineering

That duality is deeply embedded throughout this project.

There is also a more personal origin story behind the “Zilla” naming convention that appears throughout the environment.

Long before ZillaSIEM™ or ZillaAI existed, there was:

Hamzilla

Years ago, I became heavily involved in Software Defined Radio (SDR) and software-defined audio processing for amateur radio systems. At the time, many of the digital signal processing workloads were computationally demanding, especially when running multiple SDR applications, audio-routing systems, filtering chains, and visualization tools simultaneously.

That led to building increasingly powerful workstation-class systems optimized for:

- CPU performance
- memory capacity
- graphics acceleration
- multitasking
- continuous runtime stability

The naming convention evolved naturally:

Hamzilla = Ham Radio Beast PC

A slightly ridiculous name, admittedly — but it stuck.

What is interesting in hindsight is how well those systems aged. Across roughly fifteen years, only a handful of these “Hamzilla” systems were built, yet each one delivered years of operational usefulness far beyond their original purpose.

The current generation, originally designed years ago for SDR and compute-heavy radio workflows, eventually evolved into:

- a Linux infrastructure server

- an AI experimentation platform
- a SIEM runtime host
- a telemetry-processing environment
- a local inference platform
- an operational engineering system

With a still top notch AMD CPU, DDR5 128GB Memory, RTX 4090 24GB OC, super fast Storage, it rocks AI! SDR Zone Picture below!

What once powered digital radio experimentation is now powering local AI orchestration and security telemetry engineering.

That evolution feels strangely appropriate.

This project also reinforced something important:

Meaningful operational visibility no longer requires enterprise budgets, dedicated SOC facilities, or massive infrastructure investments. Modern open-source tooling, local AI models, containerized workflows, and commodity hardware have dramatically lowered the barrier to building sophisticated operational environments. The prototype, APi 5 with Ethernet Hat, two 2.5gb Taps, and commercial consumer Firewall, the cost was cheap!

The challenge is no longer access to tooling.

The challenge is whether defenders are willing to learn fast enough to keep up.

This white paper is therefore not intended to present ZillaSIEM™ as a perfect platform, a finished product, or a universal solution. It is a living engineering environment — one designed to explore what practical AI-assisted operational defense may look like in an era where both attackers and defenders increasingly have access to advanced automation.

Disclaimer, I have deidentified many details out of a secure mindset. This is not a blue print to hack me!

Likewise, ZillaAI itself remains heavily under active development. The orchestration ecosystem that originally gave birth to ZillaSIEM™ continues evolving into a much broader environment focused on secure local AI operations, intelligent orchestration, workflow coordination, operational continuity, and human-governed AI-assisted productivity.

That ecosystem deserves its own white paper someday.

But one lesson became abundantly clear during this journey:

"The more capable AI systems become, the more important telemetry, observability, monitoring, validation, and operational awareness become around the systems we trust them to control."

If there is a central idea behind this entire project, it is probably this:

The answer to increasingly intelligent offensive capability is not less understanding.
It is deeper visibility, faster learning, better tooling, and more adaptive defense.

That belief is what ultimately led to ZillaSIEM™.

Hamzilla lower right, LabZilla lower left! ***"Man did I ever have fun!"***

Table of Contents

1. Executive Summary

ZillaSIEM™ is a local-first Security Information and Event Management (SIEM) and Network Detection and Response (NDR) platform designed to support telemetry collection, operational visibility, threat enrichment, detection engineering, and analyst review workflows within a controlled private-network environment.

The platform was developed to address a practical operational challenge: creating a unified telemetry and analysis environment capable of aggregating network, infrastructure, and runtime signals while remaining operator-controlled, modular, and adaptable to evolving workflows. Rather than relying exclusively on externally hosted SaaS monitoring platforms, ZillaSIEM™ emphasizes local telemetry ownership, trusted-LAN deployment, structured evidence retention, and analyst-oriented review pipelines.

The architecture combines multiple open-source and custom-developed components, including Loki for log and event storage, Postgres for structured evidence persistence, Grafana for dashboards and alerting, Suricata-derived telemetry for network visibility, and multiple Python-based processing workers responsible for enrichment, attribution, and operational analysis.

A major design objective of the project was to separate telemetry processing from operator orchestration. ZillaSIEM™ owns telemetry collection, enrichment, evidence generation, dashboards, and alerting, while ZillaAI provides the higher-level operator experience, summaries, workflow coordination, AI-assisted review surfaces, and analyst interaction layers. This separation improves maintainability, reduces coupling between operational tooling and telemetry logic, and establishes cleaner ownership boundaries between runtime components.

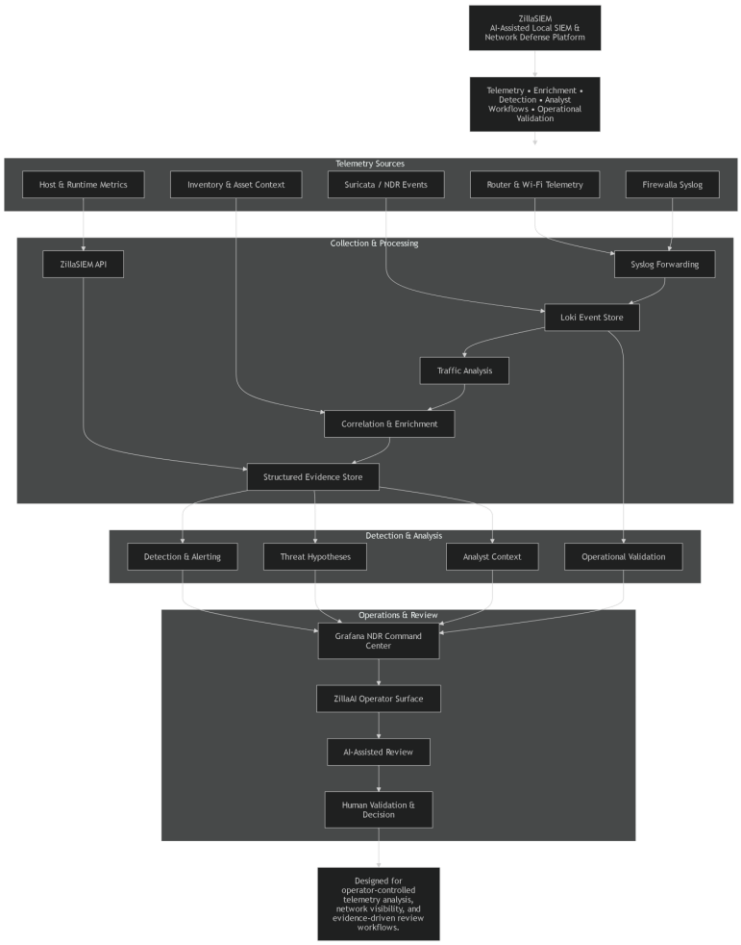
The project also explored the practical use of AI-assisted engineering workflows. Codex was used as a bounded engineering assistant to support repository inspection, implementation generation, deployment preparation, troubleshooting, operational documentation, and runtime validation. Human approval and operational control were retained for all high-impact actions including deployment, exposure decisions, infrastructure changes, and runtime validation activities. The resulting workflow demonstrated how AI-assisted engineering can accelerate implementation while preserving operator governance and review.

Operationally, the platform supports:

- Local telemetry ingestion
- Structured event enrichment
- Threat hypothesis generation
- Dashboard-driven operational monitoring
- Detection and alerting workflows
- Source identity correlation
- Network traffic analysis
- Runtime health validation
- Evidence-oriented analyst review
- EDR Intelligence ingestion

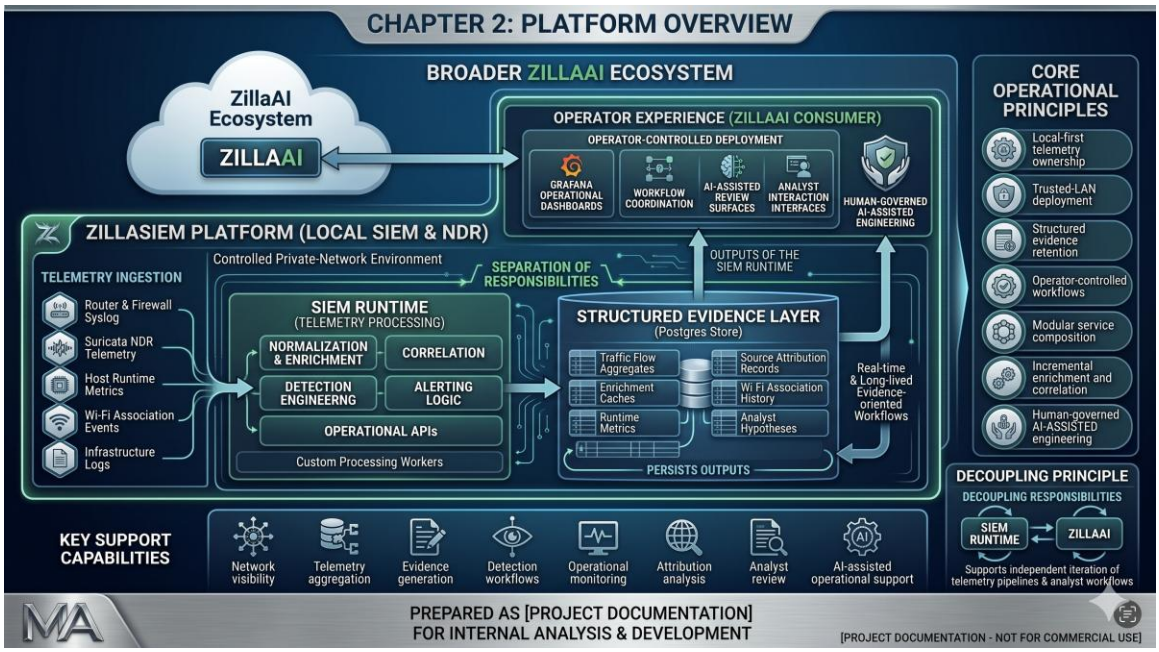
The environment intentionally avoids broad public exposure and instead emphasizes trusted-LAN operation, loopback-oriented internal services, bounded ingest paths, and operationally constrained access patterns. This security posture reflects the project's focus on controlled deployment and evidence-driven monitoring rather than large-scale external service exposure.

The whitepaper documents the architecture, operational workflows, telemetry pipelines, detection logic, enrichment mechanisms, trust boundaries, persistence strategies, and engineering methodology used to build and validate the ZillaSIEM™ platform. It is intended as both an engineering reference and an operational architecture overview for future development, maintenance, and review activities.



2. Platform Overview

ZillaSIEM™ was designed as a modular local SIEM and network defense platform focused on operational telemetry visibility, evidence-oriented analysis, and analyst workflow support within a trusted private-network environment. The platform integrates telemetry collection, structured evidence generation, enrichment pipelines, detection engineering, and dashboard-driven operational review into a unified architecture intended for operator-controlled deployment and iterative development.



The platform operates within the broader ZillaAI ecosystem while maintaining clear separation of responsibilities between telemetry processing and operator interaction layers. ZillaSIEM™ owns the telemetry runtime, including collectors, enrichment workers, dashboards, alerting logic, structured evidence storage, and operational APIs. ZillaAI consumes the outputs of the SIEM runtime and provides the higher-level operator experience, workflow coordination, AI-assisted review surfaces, and analyst interaction interfaces.

This separation was intentionally designed to establish clean ownership boundaries between telemetry infrastructure and operational orchestration. By decoupling these responsibilities, the environment supports independent iteration of telemetry pipelines, analyst workflows, dashboards, and AI-assisted operational features without tightly coupling all functionality into a single monolithic application.

The platform architecture emphasizes several core operational principles:

- Local-first telemetry ownership
- Trusted-LAN deployment
- Structured evidence retention
- Operator-controlled workflows
- Modular service composition
- Incremental enrichment and correlation
- Human-governed AI-assisted engineering

Telemetry enters the platform through multiple collection paths, including router and firewall syslog streams, Suricata-derived NDR telemetry, host runtime metrics, Wi-Fi association events, and operational infrastructure logs. These telemetry streams are normalized, enriched, correlated, and converted into structured evidence suitable for operational review and detection workflows.

Loki serves as the primary event and log storage layer for high-volume telemetry ingestion and query operations. Postgres functions as the structured evidence store for aggregated traffic data, enrichment caches, runtime metrics, source attribution records, Wi-Fi association history, and analyst hypotheses. Grafana provides the operational dashboard and alerting layer, while custom processing workers perform enrichment, correlation, attribution, and analysis functions.

The architecture supports both real-time operational review and longer-lived evidence-oriented workflows. Rather than relying exclusively on transient dashboard interpretation, the platform persists analyst-relevant outputs such as traffic hypotheses, destination dispositions, attribution context, and correlation results into structured relational tables. This design supports continuity of investigation, operational review, and future workflow automation.

A major operational objective of the environment is preserving visibility while minimizing unnecessary exposure. Internal services such as Loki and Postgres remain loopback-oriented wherever possible, while trusted-LAN access to dashboards and APIs is intentionally bounded through host firewall controls and restricted network exposure. Public-facing exposure of ingest interfaces and telemetry internals is intentionally avoided.

The environment also incorporates operational validation mechanisms intended to improve runtime reliability and troubleshooting visibility. These include health checks, telemetry freshness monitoring, container state validation, dashboard status verification, and post-reboot operational checks designed to ensure telemetry continuity and service availability after deployment or infrastructure maintenance activities.

From an engineering perspective, the platform was developed through an iterative AI-assisted workflow. Codex supported repository inspection, implementation generation, deployment preparation, troubleshooting assistance, and documentation generation while operational control and approval authority remained human-governed. This workflow allowed rapid iteration while preserving review discipline and operational accountability.

ZillaSIEM™ is therefore not simply a collection of dashboards or log viewers. It represents an integrated operational telemetry platform designed to support:

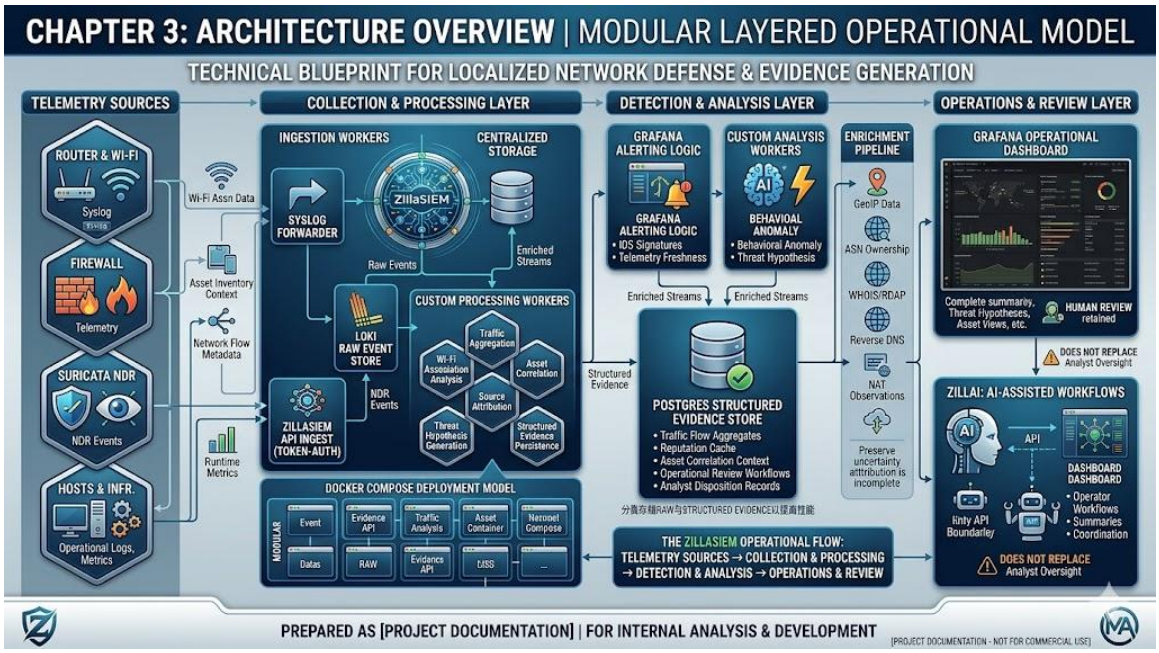
- Network visibility
- Telemetry aggregation
- Evidence generation

- Detection workflows
- Operational monitoring
- Attribution analysis
- Analyst review
- AI-assisted operational support

within a controlled and modular architecture suitable for continued expansion and refinement.

3. Architecture Overview

The ZillaSIEM™ architecture was designed around a layered operational model intended to separate telemetry ingestion, event processing, structured evidence generation, detection workflows, and analyst interaction into modular and independently maintainable components. The resulting environment supports operational visibility and network defense workflows while preserving local deployment control, bounded exposure, and evidence-oriented analysis.



At a high level, the platform architecture follows the flow:

Telemetry Sources → Collection & Processing → Detection & Analysis → Operations & Review

This layered approach allows telemetry to move from raw event generation through normalization, enrichment, attribution, structured evidence persistence, detection logic, and ultimately analyst-facing operational review surfaces.

Telemetry Sources

The environment aggregates telemetry from multiple operational sources across the local infrastructure environment. These telemetry sources include:

- Router and Wi-Fi syslog events
- Firewall telemetry
- Suricata NDR events
- Host operational logs
- Runtime metrics
- Asset inventory context
- Wi-Fi association data
- Network flow metadata

The telemetry design intentionally combines both network-centric and infrastructure-centric visibility in order to improve contextual analysis and source attribution. Rather than treating network traffic in isolation, the platform attempts to correlate traffic observations with asset identity, inventory state, Wi-Fi associations, NAT observations, and infrastructure health information.

Collection And Processing Layer

Telemetry collection and processing responsibilities are distributed across multiple modular services and workers.

Router and firewall logs are forwarded through the network syslog forwarder into Loki for centralized event storage and query access. Suricata-derived NDR telemetry is similarly ingested into Loki through the local event pipeline. Runtime metrics from Windows systems are submitted through a token-authenticated ZillaSIEM™ API ingest path rather than through direct database access.

Loki functions as the primary raw event and telemetry store. It provides the event-query layer used by enrichment workers, dashboard queries, stream validation checks, and operational review functions.

Custom processing workers continuously analyze and enrich telemetry streams. These workers perform tasks such as:

- Traffic aggregation
- Destination enrichment
- Wi-Fi association analysis
- Source attribution
- Asset correlation
- Threat hypothesis generation
- Structured evidence persistence

This processing layer intentionally converts high-volume raw telemetry into smaller, structured, analyst-oriented evidence records suitable for operational review and long-term workflow continuity.

Structured Evidence Layer

Postgres serves as the structured evidence and operational context store for the platform. While Loki retains high-cardinality event telemetry, Postgres stores the enriched and relational outputs generated by the processing pipeline.

Structured evidence includes:

- Traffic flow aggregates
- Destination ownership information
- Reputation cache data
- Runtime metrics
- Wi-Fi association observations
- Asset correlation context
- Threat hypotheses
- Analyst-oriented disposition records

This separation between raw event storage and structured evidence persistence allows dashboards and analyst workflows to operate against stable relational data rather than requiring repeated expensive queries against high-volume raw telemetry streams.

The evidence-oriented model also improves investigation continuity by preserving attribution context, analyst hypotheses, and enrichment results across dashboard refreshes and operational review sessions.

Detection And Analysis Layer

Detection and analysis responsibilities are distributed across both Grafana alerting logic and custom analysis workers.

Detection logic includes:

- IDS signature analysis
- Telemetry freshness monitoring
- Infrastructure degradation checks
- Behavioral anomaly detection
- Threat hypothesis generation
- Destination disposition analysis

The analysis pipeline combines telemetry context with enrichment information including:

- GeoIP data
- ASN ownership
- WHOIS/RDAP records
- Reverse DNS
- Inventory state
- NAT observations
- Wi-Fi association evidence
- Optional reputation data

This layered enrichment process improves attribution accuracy and helps distinguish expected operational traffic from suspicious, unresolved, or anomalous behavior.

The architecture intentionally preserves uncertainty where attribution is incomplete rather than silently suppressing unresolved observations. Unknown devices, unattributed flows, or unresolved destinations remain visible to support operational review and future investigation.

Operations And Review Layer

Grafana provides the primary operational dashboard and alerting interface for the environment. Dashboards present:

- Network traffic summaries
- Destination analysis
- NDR event visibility
- Runtime metrics
- Asset inventory views
- Telemetry freshness
- Detection alerts
- Threat hypotheses
- Operational health information

ZillaAI consumes outputs from ZillaSIEM™ through API boundaries and dashboard integrations rather than directly owning telemetry processing logic. This separation allows ZillaAI to provide operator-facing workflows, summaries, AI-assisted analysis surfaces, and workflow coordination while ZillaSIEM™ remains responsible for telemetry and evidence generation.

The operational workflow model intentionally retains human review and operator validation as part of the analysis process. AI-assisted workflows support operational efficiency and evidence review but do not replace analyst oversight or human decision-making.

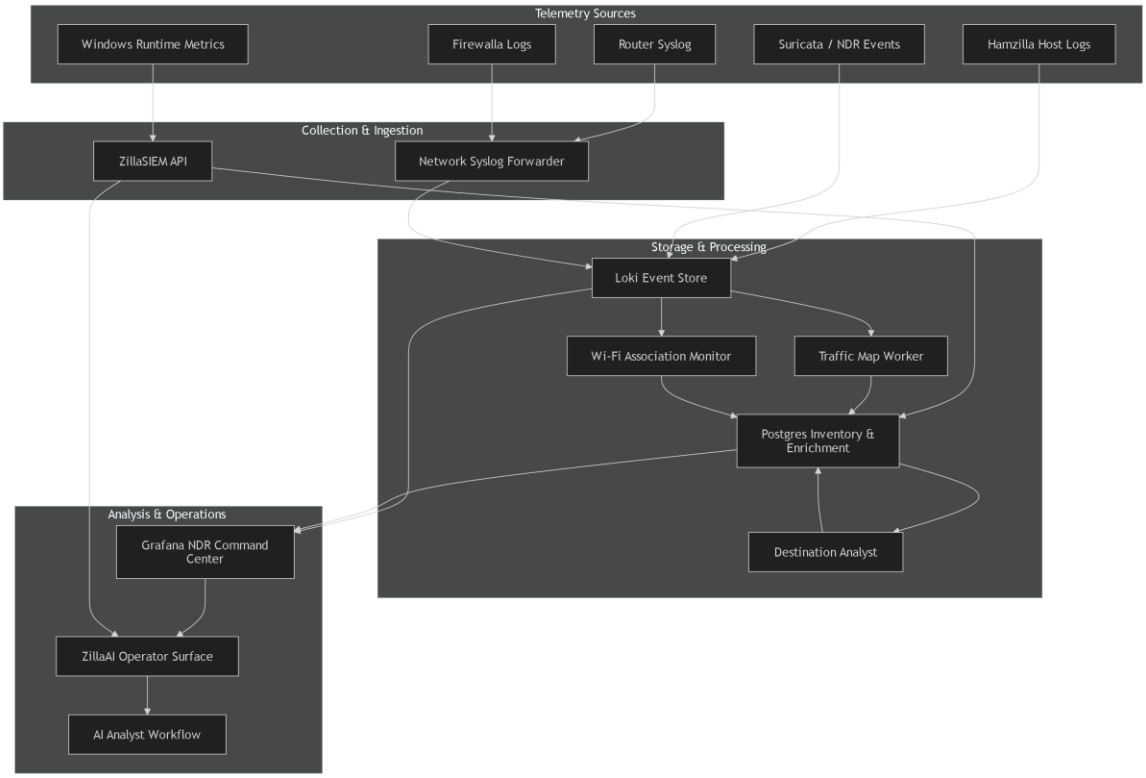
Deployment Model

The environment is deployed through Docker Compose on the Hamzilla runtime host. Services are modularized into separate containers responsible for:

- Event forwarding
- Evidence APIs
- Traffic analysis
- Wi-Fi association monitoring
- Threat analysis
- Dashboarding
- Storage
- Runtime monitoring

Internal services remain loopback-oriented wherever possible, while trusted-LAN access is selectively exposed for operational review and dashboard access. Public exposure of telemetry internals and ingest services is intentionally minimized.

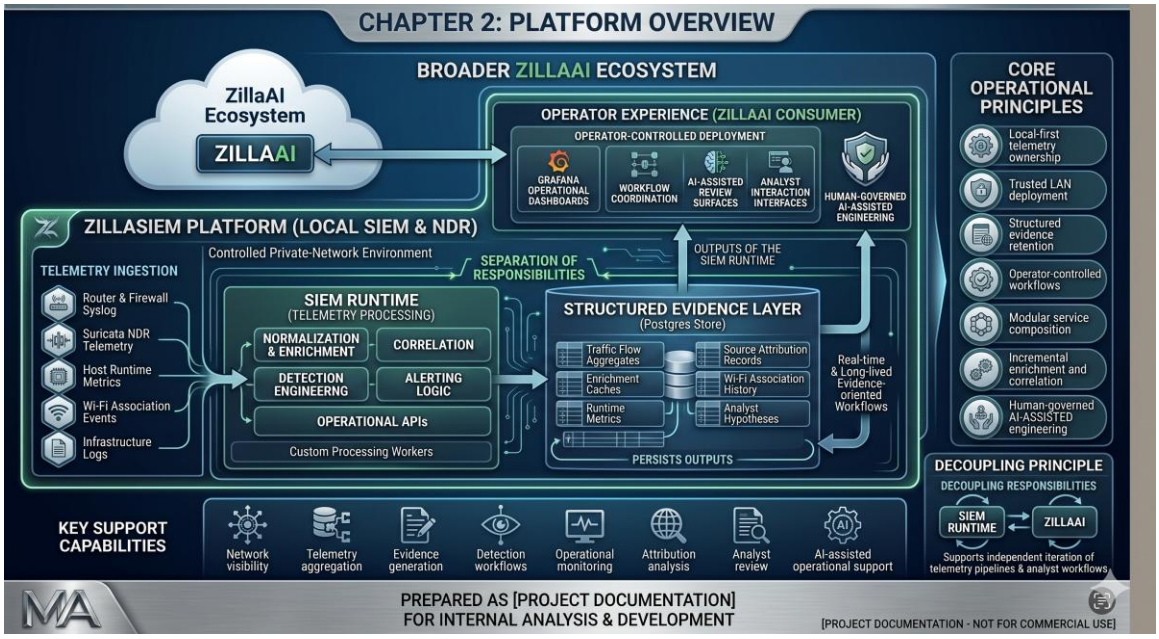
The architecture therefore reflects a balance between operational visibility, modular engineering, bounded exposure, evidence persistence, and maintainable telemetry workflows within a local-first deployment model.



ZillaSIEM™ Platform Architecture

4. Telemetry & Data Pipeline

The ZillaSIEM™ telemetry and data pipeline was designed to support continuous operational visibility, structured evidence generation, and analyst-oriented review workflows within a local-first SIEM and network defense architecture. The pipeline transforms raw telemetry streams into enriched, correlated, and operationally useful evidence while preserving separation between high-volume event storage and structured analytical data.



The telemetry design intentionally combines network-centric telemetry, infrastructure telemetry, runtime metrics, and inventory context into a unified processing model. This approach improves operational visibility and supports more accurate source attribution, traffic analysis, and detection workflows than relying on isolated log streams alone.

At a high level, the telemetry pipeline follows the sequence:

Telemetry Collection → Event Storage → Enrichment & Correlation →

Structured Evidence Generation → Detection & Analysis → Operational Review

Telemetry Sources

The environment ingests telemetry from several operational sources across the trusted network environment.

Primary telemetry sources include:

- Router and Wi-Fi syslog events
- Firewall telemetry
- Suricata EVE network telemetry
- Host operational logs
- Endpoint runtime metrics
- Asset inventory context
- Wi-Fi association events

- Traffic flow metadata

These telemetry streams collectively provide visibility into:

- Network traffic behavior
- Device activity
- Infrastructure health
- Runtime state
- Asset attribution
- Detection events
- Operational anomalies

The architecture intentionally preserves both raw telemetry and enriched evidence in order to support operational review, troubleshooting, and future investigation workflows.

Syslog Collection Pipeline

Router and firewall telemetry are forwarded into the platform using a dedicated network syslog forwarder service. The forwarder receives trusted syslog traffic and normalizes labels prior to ingestion into Loki.

The syslog pipeline provides visibility into:

- Network events
- Firewall activity
- Wi-Fi association behavior
- Device connection state
- Operational infrastructure logs
- Router health events

Separating syslog forwarding into its own service improves modularity and allows ingestion behavior to evolve independently from downstream enrichment and dashboard logic.

The syslog pipeline was designed around trusted-LAN assumptions and intentionally avoids broad external exposure.

Suricata / NDR Event Pipeline

Network Detection and Response telemetry is generated from Suricata EVE streams within the NDR tap path. These telemetry streams provide visibility into network behavior at the flow and protocol-analysis level.

Observed event types include:

- Flow records
- Alert events
- DNS telemetry
- TLS events
- QUIC traffic
- HTTP events
- Packet drop information

Suricata telemetry is ingested into Loki and subsequently consumed by downstream enrichment and traffic-analysis workers.

The platform uses Loki selectors and event-type filtering to separate:

- NDR telemetry
- Infrastructure logs
- Router telemetry
- Host operational events

This separation improves query efficiency and allows downstream processing workers to focus on specific telemetry categories.

Runtime Metrics Pipeline

Runtime metrics from Windows systems are ingested through the ZillaSIEM™ API using token-authenticated submission workflows. This ingest path was intentionally designed to avoid direct database exposure or unrestricted write access from endpoint systems.

The runtime metrics pipeline supports ingestion of:

- Host status data
- Runtime health metrics
- Operational telemetry

- Endpoint state information

Submitted metrics are normalized and persisted into structured Postgres tables for dashboard presentation and operational review.

This ingest model establishes a bounded telemetry submission path while preserving local operational visibility.

Loki Event Storage Layer

Loki serves as the primary raw event and telemetry storage layer within the environment.

Loki was selected because it supports:

- Efficient event ingestion
- Label-based querying
- Log stream separation
- Dashboard integration
- Lightweight local deployment
- Operational scalability for local telemetry workflows

Raw telemetry retained in Loki includes:

- Syslog streams
- NDR events
- Infrastructure logs
- Runtime operational logs
- Stream validation telemetry

The platform intentionally uses Loki for high-cardinality event storage while delegating structured evidence persistence to Postgres.

This separation allows the environment to preserve operational visibility without forcing dashboards and analyst workflows to depend exclusively on repeated queries against large raw telemetry datasets.

Enrichment And Correlation Pipeline

Several custom workers continuously process telemetry stored in Loki in order to generate structured operational evidence.

The enrichment and correlation pipeline performs functions including:

- Traffic aggregation
- Source attribution
- Destination enrichment
- Wi-Fi association analysis
- NAT correlation
- Asset inventory matching
- Threat hypothesis generation
- Reputation caching

Traffic and destination enrichment may include:

- GeoIP lookups
- ASN ownership
- WHOIS/RDAP context
- Reverse DNS resolution
- Inventory relationships
- Optional reputation information

The correlation pipeline attempts to preserve contextual visibility rather than aggressively suppressing uncertain or unresolved telemetry. Unknown devices, unattributed flows, and unresolved destinations remain visible for analyst review rather than being silently normalized into assumed-safe states.

This design supports operational transparency and evidence-oriented analysis.

Structured Evidence Persistence

Processed telemetry and enriched evidence are persisted into Postgres using structured relational tables.

Examples of persisted evidence include:

- Traffic flow aggregates
- Destination enrichment caches
- Runtime metrics

- Wi-Fi association history
- Source attribution context
- Threat hypotheses
- Analyst-oriented dispositions

The structured evidence layer improves dashboard stability and operational continuity by allowing analysts and dashboards to operate against normalized relational data rather than repeatedly reconstructing evidence from raw event streams.

The persistence model also supports longer-lived operational workflows by preserving analyst-relevant outputs across dashboard refreshes, worker restarts, and infrastructure maintenance activities.

Detection And Operational Review

Processed telemetry and structured evidence ultimately feed the platform's detection and analyst review workflows.

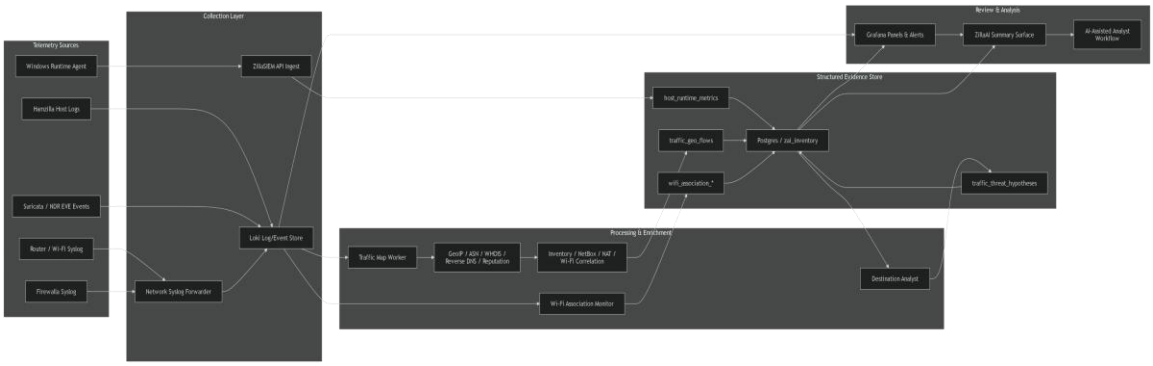
Detection logic evaluates telemetry for:

- IDS signature activity
- Telemetry degradation
- Infrastructure failures
- Behavioral anomalies
- Suspicious destinations
- Threat hypotheses
- Operational drift

Outputs are presented through:

- Grafana dashboards
- Alerting workflows
- ZillaAI summaries
- AI-assisted review surfaces
- Operational drilldowns

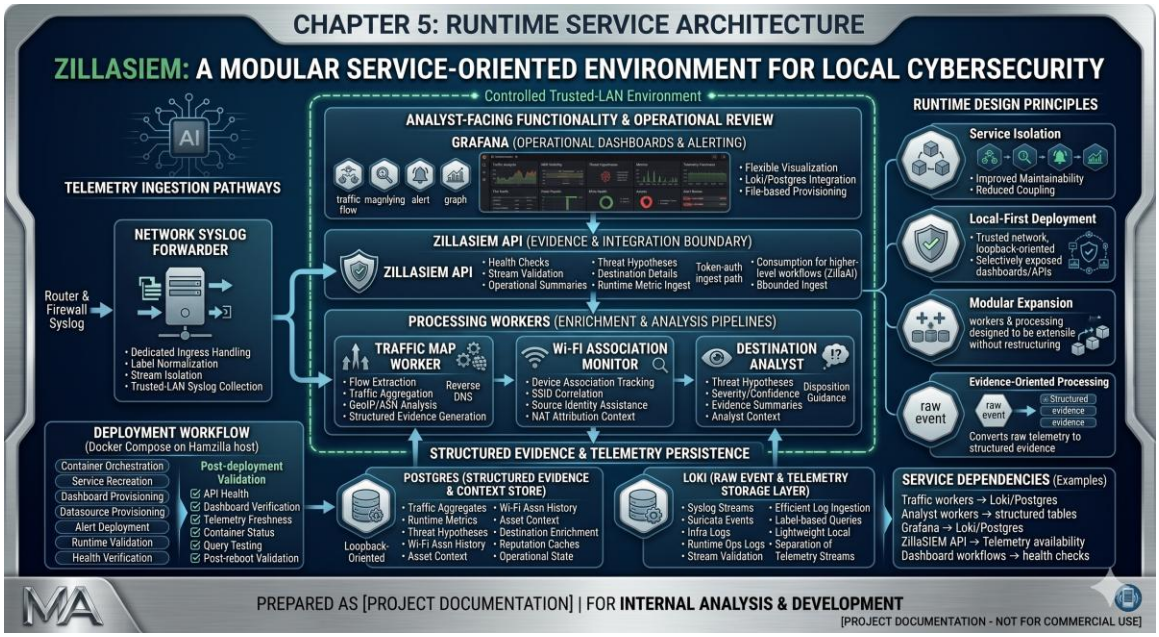
The telemetry pipeline therefore functions not simply as a logging mechanism, but as a continuous operational evidence-generation system intended to support visibility, attribution, investigation, and analyst review within the broader ZillaSIEM™ architecture.



ZillaSIEM™ Telemetry & Evidence Pipeline

5. Runtime Service Architecture

The ZillaSIEM™ runtime architecture was designed as a modular service-oriented environment intended to separate telemetry collection, enrichment, evidence generation, operational analysis, and analyst-facing functionality into independently maintainable components. The runtime model prioritizes operational clarity, service isolation, maintainability, and local deployment control while supporting continuous telemetry processing and dashboard-driven analysis workflows.



The environment is orchestrated through Docker Compose on the Hamzilla runtime host. Individual services are deployed as isolated containers responsible for specific operational functions such as telemetry forwarding, enrichment, evidence APIs, traffic analysis, dashboarding, or structured evidence persistence.

This modular deployment model provides several operational advantages:

- Separation of responsibilities
- Independent service lifecycle management
- Simplified troubleshooting
- Repeatable deployment workflows
- Incremental service expansion
- Clear ownership boundaries
- Operational containment of failures

The architecture intentionally avoids consolidating all telemetry and analysis logic into a single monolithic runtime.

Runtime Design Principles

Several core principles influenced the runtime service architecture:

Service Isolation

Each major operational function executes within its own service boundary wherever practical. Telemetry forwarding, evidence APIs, enrichment workers, dashboards, and structured storage are intentionally separated into distinct runtime components.

This isolation improves maintainability and reduces coupling between ingestion logic, enrichment workflows, operational dashboards, and analyst interfaces.

Local-First Deployment

The environment is designed for local deployment within a trusted network environment rather than broad cloud exposure. Internal services such as Loki and Postgres remain loopback-oriented wherever possible, while operational interfaces such as dashboards and APIs are selectively exposed only to trusted LAN clients.

Modular Expansion

Workers and processing services are designed to be independently extensible. New telemetry processors, enrichment pipelines, or analyst workflows can be added without requiring large-scale architectural restructuring.

Evidence-Oriented Processing

The runtime architecture supports the conversion of raw telemetry into structured operational evidence rather than relying exclusively on transient log review or dashboard inspection.

Core Runtime Services

The runtime environment includes several primary services responsible for telemetry collection, enrichment, storage, operational dashboards, and analyst workflows.

Postgres

Postgres functions as the structured evidence and operational context store for the environment.

The database persists:

- Traffic aggregates
- Runtime metrics
- Threat hypotheses
- Wi-Fi association history
- Asset context
- Destination enrichment
- Reputation caches
- Operational state information

Postgres was selected because it provides:

- Durable relational storage
- Flexible structured querying
- Dashboard compatibility
- Reliable local deployment
- Operational simplicity
- Structured evidence persistence

The database is intentionally loopback-oriented to minimize unnecessary exposure.

Loki

Loki serves as the primary raw event and telemetry storage layer.

Telemetry retained within Loki includes:

- Syslog streams
- Suricata events
- Infrastructure logs
- Runtime operational logs
- Stream validation telemetry

Loki supports:

- Efficient log ingestion
- Label-based queries
- Grafana integration
- Lightweight local operation
- Separation of telemetry streams

The platform intentionally separates raw event storage from structured evidence persistence by using Loki for high-cardinality telemetry and Postgres for enriched analytical outputs.

Grafana

Grafana provides the primary operational dashboard and alerting surface for the environment.

The dashboard layer includes:

- Traffic analysis views
- NDR visibility panels
- Threat hypotheses
- Runtime metrics
- Telemetry freshness monitoring
- Infrastructure health views
- Asset inventory panels

- Alert review workflows

Grafana was selected because it provides:

- Flexible visualization
- Alerting support
- Loki integration
- Postgres integration
- File-based provisioning
- Operational dashboard maturity

Provisioning for dashboards, datasources, and alerts is maintained within the ZillaSIEM™ repository to improve reproducibility and deployment consistency.

ZillaSIEM™ API

The ZillaSIEM™ API acts as the primary evidence and operational integration boundary between telemetry services and higher-level operational workflows.

The API exposes endpoints for:

- Health checks
- Stream validation
- Operational summaries
- Traffic map data
- Threat hypotheses
- Destination details
- Runtime metric ingest

This API boundary allows ZillaAI to consume SIEM outputs without directly owning telemetry processing logic or raw event-query responsibilities.

The API also supports bounded ingest workflows for endpoint runtime metrics using token-authenticated submissions.

Network Syslog Forwarder

The network syslog forwarder receives router and firewall telemetry and forwards normalized events into Loki.

This separation provides:

- Dedicated telemetry ingress handling
- Label normalization
- Stream isolation
- Trusted-LAN syslog collection
- Modular ingestion control

The syslog forwarder operates as a dedicated telemetry ingestion component rather than embedding ingestion logic into downstream services.

Processing Workers

Several processing workers continuously analyze and enrich telemetry streams.

Traffic Map Worker

The traffic map worker processes Suricata-derived telemetry from Loki and generates structured traffic flow evidence.

Functions include:

- Flow extraction
- Traffic aggregation
- Destination enrichment
- GeoIP analysis
- ASN resolution
- Reverse DNS analysis
- Structured evidence generation

Processed outputs are persisted into relational Postgres tables for dashboard consumption and operational review.

Wi-Fi Association Monitor

The Wi-Fi association monitor analyzes router telemetry in order to preserve Wi-Fi association history and source attribution context.

The worker supports:

- Device association tracking

- Churn analysis
- SSID correlation
- Source identity assistance
- NAT attribution context

This enrichment improves operational visibility into device identity and network behavior.

Destination Analyst

The destination analyst worker evaluates enriched traffic data and generates structured threat hypotheses.

The worker preserves:

- Threat hypotheses
- Severity
- Confidence
- Evidence summaries
- Analyst context
- Disposition guidance

Persisting these outputs improves continuity across operational review sessions and dashboard refreshes.

Service Dependencies

Service dependencies are explicitly modeled within the runtime architecture.

Examples include:

- Traffic workers depending on Loki and Postgres
- Analyst workers depending on structured evidence tables
- Grafana depending on Loki and Postgres datasources
- ZillaSIEM™ API depending on telemetry availability
- Dashboard workflows depending on operational health checks

Explicit dependency modeling improves startup reliability and operational consistency during deployment and restart activities.

Deployment Workflow

The runtime environment is deployed through Docker Compose using repo-managed configuration and provisioning assets.

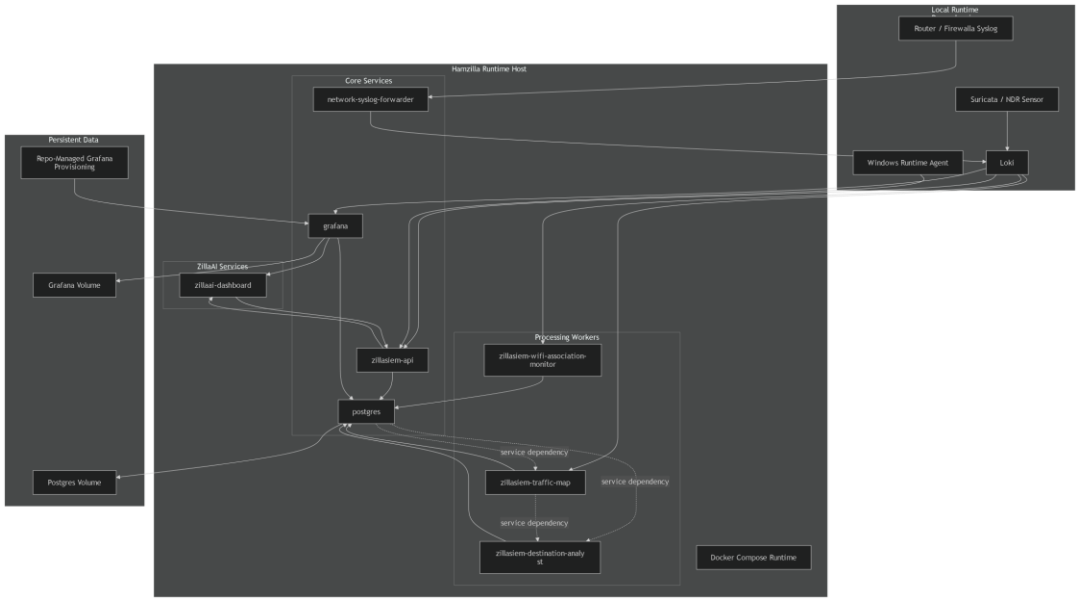
Deployment activities include:

- Container orchestration
- Service recreation
- Dashboard provisioning
- Datasource provisioning
- Alert deployment
- Runtime validation
- Health verification

Operational validation after deployment includes:

- API health checks
- Dashboard verification
- Telemetry freshness checks
- Container status validation
- Query testing
- Post-reboot validation

The runtime architecture therefore represents a modular operational telemetry platform designed to support continuous visibility, structured evidence generation, enrichment workflows, analyst review, and maintainable service-oriented deployment within a local-first operational environment.



ZillaSIEM™ Runtime Service Architecture

6. Trust Boundaries & Exposure Model

The ZillaSIEM™ environment was designed with a strong emphasis on bounded exposure, local-first deployment, and operational separation between trusted telemetry sources, internal processing components, analyst-facing services, and external networks. The resulting trust model intentionally minimizes unnecessary public exposure while preserving operational visibility and trusted-LAN accessibility for monitoring and review workflows.

Rather than assuming broad external accessibility or cloud-native exposure, the platform adopts a constrained operational posture in which telemetry processing, structured evidence generation, and internal service communication occur primarily within a protected local runtime boundary hosted on the Hamzilla system.

At a high level, the trust model separates the environment into four primary operational zones:

- Trusted telemetry sources
- Internal runtime and processing services
- Analyst and operator access surfaces
- External/public network boundaries

This layered trust model helps reduce exposure of sensitive telemetry infrastructure while preserving operational usability for dashboarding, analysis, and maintenance activities.

Trusted Telemetry Sources

Telemetry entering the environment originates from systems considered part of the trusted operational network environment.

These sources include:

- Router infrastructure
- Firewall telemetry
- Suricata NDR sensors
- Windows runtime agents
- Host operational logs
- Asset inventory systems

Telemetry from these systems is treated as operational input into the SIEM pipeline rather than as externally untrusted internet-facing data ingestion.

The architecture assumes these telemetry sources are:

- Operator-managed
- Internal-network reachable
- Operationally trusted
- Subject to local administrative control

This trust assumption allows the environment to prioritize operational visibility and local telemetry ownership while still preserving boundaries around ingest and storage components.

Internal Runtime Boundary

The primary trust boundary within the architecture is the Hamzilla local runtime environment, which hosts:

- Loki
- Postgres
- Grafana
- ZillaSIEM™ API
- Processing workers

- Syslog forwarders
- Dashboard integrations

Internal runtime services communicate through container networking and loopback-oriented access patterns wherever practical.

The architecture intentionally avoids broad exposure of:

- Raw telemetry stores
- Internal databases
- Worker services
- Loki query interfaces
- Structured evidence stores

This design minimizes unnecessary attack surface while preserving operational functionality.

Several core services are intentionally loopback-oriented, including:

- Postgres
- Loki
- Internal evidence stores

Restricting these services to local access reduces the likelihood of accidental external exposure while simplifying operational trust assumptions.

Trusted-LAN Operational Access

Certain services are intentionally accessible to trusted LAN clients in order to support operational workflows and analyst review activities.

These services may include:

- Grafana dashboards
- ZillaAI operator interfaces
- ZillaSIEM™ API endpoints
- Operational status views

Trusted-LAN exposure is intentionally bounded through:

- Host firewall rules

- Limited port exposure
- Local network segmentation
- Token-authenticated ingest workflows
- Controlled operational access patterns

The environment intentionally avoids exposing telemetry internals directly to untrusted networks or public internet clients.

Operational access therefore remains constrained to systems and users within the trusted administrative environment.

Public Network Boundary

The architecture intentionally minimizes direct public exposure.

The public internet is not treated as a primary operational access layer for the SIEM runtime. Services such as:

- Loki
- Postgres
- Worker APIs
- Internal enrichment services
- Structured evidence stores

are intentionally not exposed directly to the public internet.

This exposure model reflects the project's operational goals:

- Preserve local telemetry ownership
- Reduce unnecessary attack surface
- Maintain operator-controlled deployment
- Avoid externally accessible ingest infrastructure
- Retain bounded operational access

Where remote access or operational connectivity is required, access is intended to occur through trusted administrative workflows rather than unrestricted public exposure.

Ingest Security Model

The platform supports multiple telemetry ingest paths, each designed with bounded operational assumptions.

Syslog Ingest

Router and firewall telemetry are received through the dedicated syslog forwarder service using trusted-LAN syslog assumptions.

This ingest path intentionally assumes:

- Known telemetry sources
- Internal network transport
- Controlled infrastructure devices
- Operationally trusted emitters

The syslog service itself remains isolated from downstream enrichment and dashboard services.

Endpoint Runtime Metrics

Endpoint runtime metrics are submitted through token-authenticated API workflows rather than direct database access.

This model improves operational control by:

- Avoiding endpoint database credentials
- Establishing bounded ingest APIs
- Supporting authentication validation
- Reducing direct access to evidence stores

The API ingest workflow therefore acts as a constrained telemetry submission boundary rather than a general-purpose exposed service interface.

Operational Separation

The trust model also preserves separation between:

- Telemetry processing
- Dashboard presentation
- AI-assisted workflows
- Operational orchestration

- Runtime administration

ZillaSIEM™ remains responsible for telemetry collection, evidence generation, and detection logic, while ZillaAI consumes operational outputs through bounded APIs and dashboard integrations.

This separation reduces coupling between telemetry internals and analyst-facing workflows while improving maintainability and operational clarity.

AI-Assisted Engineering Governance

An additional trust consideration within the environment involves the AI-assisted engineering workflow used during development and maintenance activities.

Codex-assisted engineering was intentionally bounded through:

- Human approval requirements
- Operator validation
- Controlled execution paths
- Deployment review workflows
- Runtime verification procedures

High-impact operational actions such as:

- Service exposure
- Reboots
- Infrastructure changes
- Runtime deployments
- Firewall adjustments

remained subject to explicit human approval and validation.

This governance model ensured that AI-assisted workflows supported operational efficiency without bypassing operator review or deployment accountability.

Operational Security Philosophy

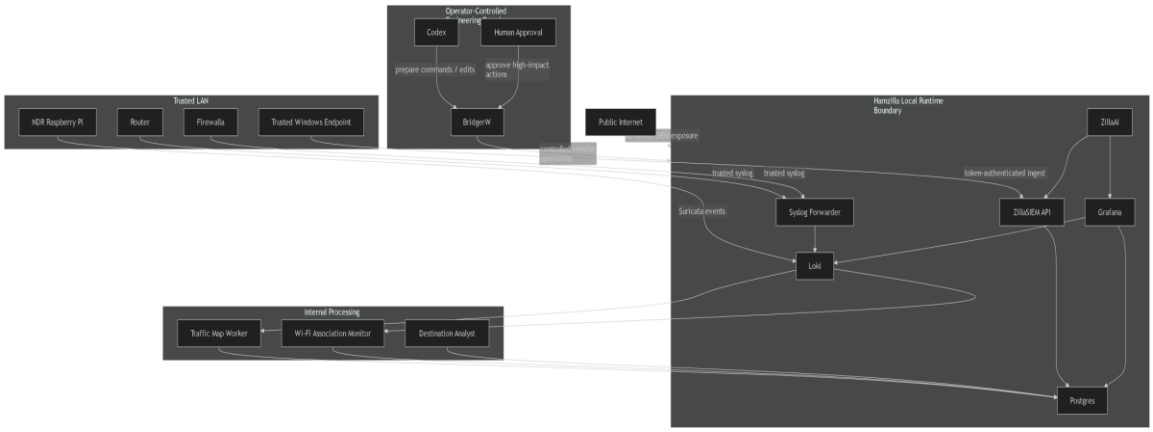
The trust and exposure model ultimately reflects several broader operational security principles:

- Local-first telemetry ownership

- Minimized external exposure
- Structured operational boundaries
- Evidence-oriented visibility
- Trusted-LAN operational access
- Bounded ingest workflows
- Human-governed operational control

Rather than prioritizing large-scale public accessibility or cloud-native deployment assumptions, the environment emphasizes operational containment, modular service boundaries, and constrained exposure suitable for a private operational telemetry platform.

The resulting architecture therefore balances visibility, maintainability, operational usability, and bounded exposure within a controlled local deployment model intended for iterative engineering and operational review workflows.



ZillaSIEM™ Trust Boundaries & Exposure Model

7. Threat Enrichment & Correlation

A primary operational objective of the ZillaSIEM™ platform is transforming raw telemetry into contextualized operational evidence suitable for analyst review, attribution analysis, detection workflows, and long-term operational visibility. To support this objective, the platform incorporates a layered enrichment and correlation pipeline responsible for associating telemetry with infrastructure context, network identity, ownership metadata, and operational disposition information.

Rather than treating telemetry solely as isolated event streams, the enrichment architecture attempts to preserve relationships between:

- Devices

- Network flows
- Infrastructure telemetry
- Asset identity
- Destination ownership
- Runtime state
- Wi-Fi associations
- Threat hypotheses

This contextual model improves operational visibility and supports more informed analysis workflows than raw log aggregation alone.

At a high level, the enrichment and correlation workflow follows the sequence:

Raw Telemetry → Source Attribution → Destination Enrichment →
Correlation → Structured Evidence → Analyst Review

Enrichment Philosophy

The enrichment pipeline was designed around several operational principles:

Preserve Context

Telemetry should retain attribution and operational context wherever possible rather than being aggressively normalized into generic event records.

Preserve Uncertainty

Unknown or unattributed observations remain visible rather than being silently suppressed or automatically classified as benign.

Separate Raw Events From Structured Evidence

Raw telemetry remains available in Loki while enriched outputs and analyst-oriented evidence are persisted into structured Postgres tables.

Incremental Enrichment

Enrichment occurs progressively through multiple workers and processing stages rather than through a single monolithic processing pipeline.

This design improves modularity, troubleshooting visibility, and maintainability.

Source Attribution

One of the primary goals of the correlation pipeline is improving attribution accuracy for observed traffic and infrastructure activity.

Observed network traffic may originate from:

- NAT environments
- Wi-Fi-connected devices
- Dynamically assigned clients
- Shared infrastructure
- Multiple endpoint classes

To improve attribution visibility, the platform correlates telemetry using:

- Wi-Fi association records
- Asset inventory data
- Router telemetry
- Runtime metrics
- Source IP observations
- Host identity mappings
- Network behavior history

The platform intentionally attempts to preserve attribution evidence rather than presenting unsupported certainty where source identity cannot be confidently resolved.

This operational approach helps distinguish between:

- Confirmed attribution
- Probable attribution
- Ambiguous attribution
- Unknown sources

which improves investigation transparency during analyst review workflows.

Wi-Fi Association Correlation

The Wi-Fi association monitor continuously analyzes router telemetry to preserve device association state and wireless network relationships.

Observed information may include:

- MAC address observations
- SSID associations
- Connection timing
- Device churn
- Association history
- Local IP relationships

This data assists with:

- Device attribution
- NAT disambiguation
- Endpoint identification
- Operational visibility
- Infrastructure analysis

Persisting Wi-Fi association history also improves continuity during investigations involving roaming devices, DHCP changes, or intermittent network activity.

Destination Enrichment

Observed network destinations are enriched using multiple contextual sources intended to improve visibility into destination ownership, geography, and operational relevance.

Enrichment workflows may include:

- GeoIP lookups
- ASN ownership resolution
- WHOIS/RDAP retrieval
- Reverse DNS analysis
- Domain context
- Reputation data
- Traffic history
- Frequency analysis

The platform intentionally separates enrichment caches from transient telemetry in order to reduce redundant external lookups and improve operational efficiency.

Examples of persisted enrichment data include:

- IP ownership records
- ASN information
- Geographic metadata
- Cached reputation observations
- Reverse DNS mappings

These enrichment results are reused across dashboard workflows, destination analysis, and threat hypothesis generation.

Traffic Aggregation

The traffic map worker processes high-volume Suricata telemetry and converts raw events into structured traffic summaries suitable for operational dashboards and analyst review.

Traffic aggregation workflows include:

- Flow summarization
- Destination grouping
- Protocol analysis
- Traffic classification
- Volume analysis
- Geographic aggregation
- Historical observation tracking

Persisting aggregated evidence reduces dashboard query complexity while improving operational continuity across sessions and deployments.

The resulting structured evidence supports:

- Traffic maps
- Destination summaries
- Protocol distribution panels
- Remote destination analysis

- Historical trend review

Threat Hypothesis Generation

The destination analyst worker evaluates enriched traffic and attribution evidence in order to generate operational hypotheses regarding observed network activity.

The platform intentionally uses the term "hypothesis" rather than asserting definitive malicious classification in situations where evidence may be incomplete or probabilistic.

Hypotheses may incorporate:

- Destination ownership
- Reputation observations
- Infrastructure context
- Protocol behavior
- Geographic relationships
- Traffic frequency
- Observed anomalies
- Attribution confidence

Generated hypotheses are persisted into structured relational tables along with:

- Severity
- Confidence
- Supporting evidence
- Analyst notes
- Disposition recommendations

This evidence-oriented model improves operational continuity and supports later review or workflow automation.

Correlation Between Data Sources

The correlation engine attempts to combine evidence from multiple telemetry sources into coherent operational context.

Correlated telemetry may include:

- Syslog events

- Suricata telemetry
- Runtime metrics
- Inventory state
- Wi-Fi association history
- Asset identity
- Destination enrichment
- Operational health signals

This layered visibility improves analyst understanding by reducing fragmentation between telemetry systems and infrastructure observations.

The architecture intentionally avoids tightly coupling enrichment logic to any single telemetry source in order to preserve flexibility and maintainability as new telemetry workflows are introduced.

Structured Evidence Persistence

Processed enrichment outputs are persisted into Postgres relational tables for operational continuity and dashboard integration.

Examples of structured evidence include:

- Traffic flow summaries
- Wi-Fi association history
- Threat hypotheses
- Destination metadata
- Reputation caches
- Runtime metrics
- Attribution context

This persistence strategy improves:

- Dashboard responsiveness
- Investigation continuity
- Historical visibility
- Workflow reuse

- Analyst context retention

rather than forcing analysts to repeatedly reconstruct evidence directly from raw event streams.

Analyst Review Integration

The enrichment and correlation pipeline ultimately supports analyst-facing operational review workflows presented through Grafana and ZillaAI.

Operational review surfaces may include:

- Traffic maps
- Threat hypotheses
- Destination summaries
- Attribution context
- Runtime health
- Telemetry freshness
- Operational anomalies

AI-assisted workflows consume enriched evidence and structured context rather than raw unprocessed telemetry alone. This improves the quality and consistency of operational summaries while preserving analyst visibility into supporting evidence.

Human review remains part of the operational workflow, particularly for ambiguous, incomplete, or potentially high-impact observations.

Operational Outcome

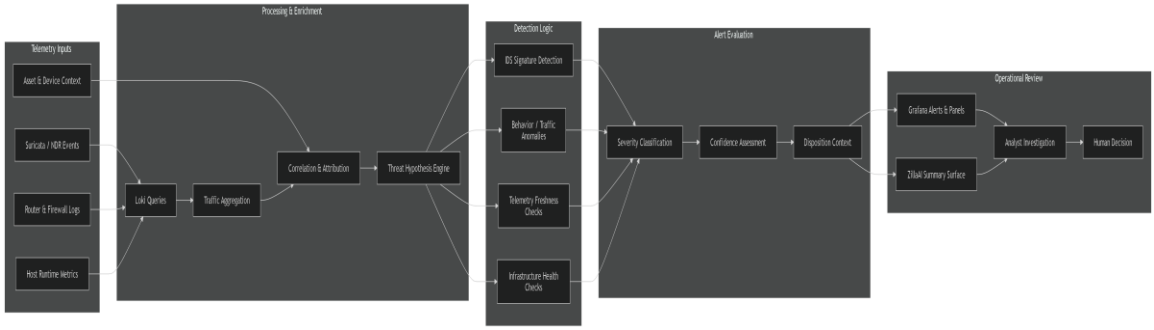
The enrichment and correlation architecture transforms the platform from a simple telemetry collection environment into an evidence-oriented operational analysis system.

Rather than merely storing logs, the platform attempts to:

- Preserve attribution context
- Correlate infrastructure observations
- Generate structured evidence
- Support analyst workflows
- Improve operational visibility

- Maintain investigation continuity

within a modular, locally controlled SIEM and network defense architecture.



ZillaSIEM™ Detection & Alerting Pipeline

8. Detection & Alerting

The ZillaSIEM™ detection and alerting architecture was designed to support operational visibility, telemetry integrity validation, anomaly identification, and evidence-oriented analyst review workflows within a locally controlled SIEM and network defense environment. Rather than focusing exclusively on traditional signature-based alerting, the platform combines multiple detection approaches intended to identify infrastructure degradation, telemetry loss, anomalous behavior, and potentially suspicious network activity while preserving contextual evidence for operational review.

The detection model intentionally balances automation with human oversight. Alerts and hypotheses are treated as operational signals requiring contextual review rather than as authoritative determinations of malicious activity. This approach reflects the platform's emphasis on evidence-oriented workflows, operational transparency, and analyst validation.

At a high level, the detection workflow follows the sequence:

Telemetry → Enrichment & Correlation → Detection Logic →
Severity & Confidence Evaluation → Analyst Review

Detection Philosophy

Several operational principles guided the detection architecture.

Preserve Analyst Context

Alerts should include supporting context, attribution information, enrichment data, and operational evidence rather than presenting isolated event notifications without explanation.

Preserve Visibility

Unknown, unattributed, or unresolved observations remain visible to analysts rather than being silently filtered or aggressively normalized into assumed-safe states.

Detect Operational Failures

Detection logic includes not only network threat observations but also telemetry degradation, ingestion failures, infrastructure instability, and operational drift conditions.

Support Human Review

AI-assisted analysis and automated logic assist operational workflows but do not replace human validation for high-impact or ambiguous observations.

Telemetry Inputs

Detection workflows consume telemetry from multiple operational sources.

Primary inputs include:

- Suricata NDR telemetry
- Router and firewall logs
- Runtime metrics
- Traffic aggregation records
- Asset inventory context
- Wi-Fi association history
- Threat enrichment data
- Structured operational evidence

This multi-source approach improves operational visibility by correlating infrastructure observations with traffic behavior and attribution context.

Signature-Based Detection

Suricata-generated alerts provide the primary signature-based detection capability within the environment.

Observed alert categories may include:

- Known malicious traffic indicators
- Suspicious protocol activity

- DNS anomalies
- TLS irregularities
- Traffic policy violations
- Exploit signatures
- Reconnaissance patterns

These alerts are retained within Loki and may also contribute to downstream enrichment and analyst workflows.

The architecture intentionally preserves raw signature observations alongside enriched evidence rather than discarding underlying telemetry after alert generation.

Behavioral & Anomaly Detection

In addition to signature-based detection, the platform incorporates operational anomaly detection workflows intended to identify unusual or potentially suspicious conditions that may not trigger traditional IDS signatures.

Examples include:

- Unexpected destination behavior
- Unusual protocol distributions
- Traffic spikes
- Unknown remote destinations
- Device churn anomalies
- Telemetry disappearance
- Infrastructure degradation
- Runtime instability

Behavioral observations are evaluated in the context of:

- Historical activity
- Device identity
- Inventory context
- Wi-Fi association state
- Destination ownership

- Geographic patterns
- Operational baselines

This layered approach improves contextual understanding while reducing reliance on isolated signature events alone.

Telemetry Freshness Monitoring

Operational visibility depends heavily on telemetry continuity. As a result, the platform treats telemetry freshness and ingestion health as important operational detection categories.

Freshness monitoring evaluates:

- Syslog continuity
- Suricata event flow
- Runtime metric updates
- Worker execution cadence
- Dashboard datasource responsiveness
- API availability

Detection logic may generate alerts for:

- Missing telemetry streams
- Stale event pipelines
- Delayed updates
- Worker failures
- Datasource degradation
- Dashboard inconsistencies

This operational focus helps identify failures in the monitoring environment itself rather than only monitoring external traffic behavior.

Infrastructure Health Detection

Infrastructure health monitoring forms an additional layer of the detection architecture.

Operational health checks may evaluate:

- Container runtime state

- Docker service availability
- API responsiveness
- Postgres connectivity
- Loki query health
- Dashboard provisioning state
- Service restart behavior

These checks improve deployment reliability and support operational troubleshooting workflows.

The platform intentionally treats infrastructure degradation as an operationally relevant event category alongside security-oriented telemetry.

Threat Hypothesis Generation

The destination analyst worker evaluates enriched traffic evidence and generates structured threat hypotheses based on observed network activity.

Hypothesis generation incorporates:

- Destination enrichment
- Traffic history
- Reputation context
- Protocol behavior
- Geographic observations
- Attribution confidence
- Operational anomalies

Rather than asserting definitive malicious classification, the system preserves uncertainty where evidence is incomplete or probabilistic.

Generated hypotheses may include:

- Severity
- Confidence
- Analyst context
- Supporting observations

- Suggested review actions
- Operational disposition guidance

Persisting these hypotheses into structured relational tables improves continuity across operational review sessions.

Severity & Confidence Evaluation

Detection outputs are evaluated using both severity and confidence considerations.

Severity reflects the potential operational impact or relevance of an observation, while confidence reflects the strength and completeness of the supporting evidence.

Examples of factors influencing confidence may include:

- Correlated telemetry sources
- Attribution certainty
- Enrichment quality
- Reputation consistency
- Historical observations
- Supporting telemetry continuity

This separation helps prevent overstating uncertain observations while still preserving operational visibility into potentially relevant activity.

Alert Presentation & Review

Alerts and operational hypotheses are surfaced primarily through Grafana dashboards and alerting workflows.

Operational review surfaces may include:

- Alert panels
- Threat hypothesis dashboards
- Telemetry freshness views
- Runtime health panels
- Traffic summaries
- Destination analysis
- Operational drilldowns

ZillaAI consumes operational outputs through bounded APIs and dashboard integrations to support AI-assisted review workflows and operator summaries.

The platform intentionally preserves analyst access to supporting evidence rather than presenting opaque alert conclusions without context.

Human Validation

Human review remains an intentional part of the detection architecture.

Analysts and operators may:

- Review supporting telemetry
- Validate attribution assumptions
- Evaluate confidence levels
- Investigate ambiguous observations
- Tune detection logic
- Adjust enrichment workflows
- Refine dashboard views

This operational model reflects the project's emphasis on evidence-oriented workflows and bounded AI assistance rather than fully autonomous decision-making.

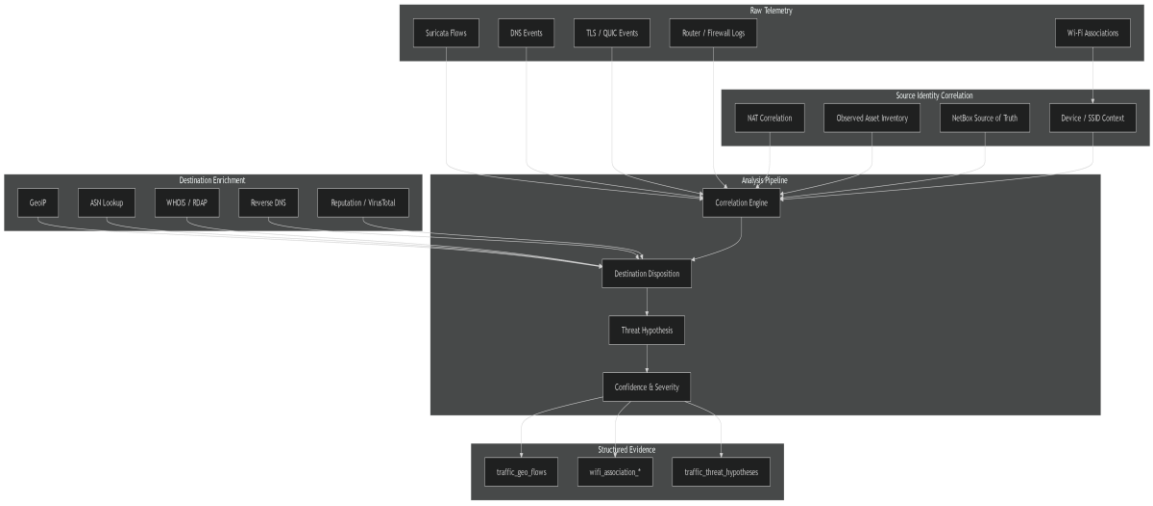
Detection Workflow Outcomes

Detection workflows may ultimately result in:

- Continued monitoring
- Follow-up investigation
- Dashboard tuning
- Detection refinement
- Infrastructure remediation
- Telemetry troubleshooting
- Evidence preservation
- Operational documentation

The platform therefore treats detection not as a standalone alert-generation mechanism, but as part of a broader operational visibility and analyst review pipeline intended to

support informed decision-making within a locally controlled SIEM and network defense environment.



ZillaSIEM™ Threat Enrichment & Correlation Pipeline

9. Analyst Workflow

The ZillaSIEM™ analyst workflow architecture was designed to support evidence-oriented operational review, telemetry interpretation, detection validation, and investigation continuity within a locally controlled SIEM and network defense environment. The workflow model intentionally combines dashboards, structured evidence, enrichment context, AI-assisted summaries, and human validation into a layered operational review process rather than relying exclusively on isolated alert notifications or raw log inspection.

The platform's analyst workflow philosophy emphasizes:

- Context preservation
- Evidence visibility
- Incremental investigation
- Operational continuity
- Human validation
- Bounded AI assistance

Rather than presenting alerts as final conclusions, the workflow architecture treats detections and hypotheses as operational signals that require contextual review and analyst interpretation.

At a high level, the analyst workflow follows the sequence:

Detection Signal → Evidence Review → Contextual Analysis →
Threat Hypothesis → Analyst Validation → Operational Outcome

Workflow Objectives

Several operational objectives influenced the design of the analyst workflow architecture.

Preserve Supporting Evidence

Analysts should be able to access the telemetry, enrichment, attribution data, and operational context supporting a detection or hypothesis.

Preserve Investigation Continuity

Operational findings, hypotheses, and evidence summaries should persist across dashboard refreshes, service restarts, and review sessions.

Reduce Context Fragmentation

Telemetry, enrichment, infrastructure state, and attribution evidence should be correlated into unified operational views wherever practical.

Support Incremental Review

Analysts should be able to move from high-level summaries into progressively deeper operational detail without losing contextual continuity.

Preserve Human Oversight

AI-assisted workflows support investigation efficiency and summarization but do not replace operator validation or human decision-making.

Operational Review Surfaces

The analyst workflow relies on multiple operational review surfaces presented through Grafana dashboards and ZillaAI interfaces.

These review surfaces include:

- Traffic maps
- Alert panels
- Runtime health dashboards
- Threat hypothesis summaries
- Destination analysis views
- Telemetry freshness panels

- Asset inventory views
- Infrastructure status panels
- Wi-Fi correlation dashboards
- Operational drilldowns

The platform intentionally presents both high-level operational summaries and low-level supporting evidence in order to support flexible investigation depth.

Detection Signal Intake

Analyst workflows may begin from several operational triggers, including:

- IDS signature alerts
- Traffic anomalies
- Telemetry freshness failures
- Infrastructure degradation alerts
- Suspicious destination observations
- Threat hypotheses
- Runtime instability
- Dashboard anomalies

These triggers are treated as investigation entry points rather than final determinations of malicious activity.

The architecture intentionally preserves uncertainty where attribution or evidence remains incomplete.

Evidence Review Workflow

Once a detection signal or operational anomaly is identified, the workflow shifts into evidence review.

Evidence review may include:

- Raw Loki event inspection
- Traffic flow review
- Destination enrichment analysis
- Runtime metric validation

- Wi-Fi association review
- Source attribution analysis
- Historical observation comparison
- Infrastructure context evaluation

This layered review model allows analysts to evaluate:

- What occurred
- Which systems were involved
- How attribution was derived
- Whether telemetry is complete
- Whether enrichment data supports escalation
- Whether infrastructure instability may explain anomalies

The workflow intentionally avoids presenting detections without access to supporting evidence.

Threat Hypothesis Evaluation

The destination analyst worker generates structured hypotheses regarding observed network activity based on enrichment and telemetry correlation.

Analyst review of hypotheses may consider:

- Severity
- Confidence
- Attribution quality
- Destination ownership
- Historical behavior
- Protocol usage
- Geographic observations
- Reputation data
- Supporting telemetry continuity

The platform intentionally uses the term "hypothesis" to preserve analytical caution where evidence may remain incomplete or probabilistic.

Analysts retain responsibility for validating or dismissing operational conclusions.

AI-Assisted Review Workflows

ZillaAI consumes operational outputs from ZillaSIEM™ in order to support AI-assisted review workflows and operational summarization.

AI-assisted workflows may assist with:

- Event summarization
- Context aggregation
- Threat explanation
- Operational narrative generation
- Dashboard interpretation
- Investigation guidance
- Workflow continuity

However, the architecture intentionally preserves analyst access to supporting evidence and operational drilldowns rather than replacing review with opaque AI-generated conclusions.

The workflow model treats AI assistance as an operational acceleration layer rather than an autonomous security decision-maker.

Operational Drilldowns

The analyst workflow supports movement between high-level operational dashboards and deeper investigation views.

Drilldown workflows may include transitions between:

- Traffic maps
- Destination summaries
- Raw event streams
- Threat hypotheses
- Runtime health panels

- Attribution records
- Wi-Fi correlation views
- Asset inventory context

This layered operational visibility improves investigation continuity while reducing fragmentation between telemetry systems and infrastructure context.

Human Validation & Decision-Making

Human validation remains a core operational principle within the workflow architecture.

Analysts and operators may:

- Validate attribution assumptions
- Review supporting telemetry
- Assess enrichment quality
- Determine operational significance
- Tune detection logic
- Escalate investigations
- Preserve evidence
- Document operational findings

This governance model reflects the project's emphasis on bounded AI assistance, evidence-oriented workflows, and operator-controlled decision-making.

Operational Outcomes

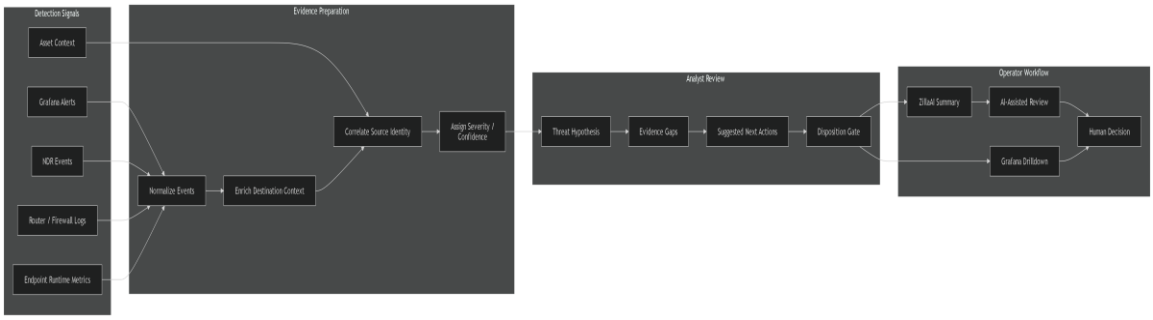
Analyst review workflows may ultimately result in several operational outcomes, including:

- Continued monitoring
- Follow-up investigation
- Dashboard refinement
- Detection tuning
- Infrastructure remediation
- Evidence documentation
- Attribution updates

- Telemetry troubleshooting

The workflow architecture therefore supports not only detection visibility but also operational continuity, contextual investigation, and evidence-driven decision-making across the broader ZillaSIEM™ platform.

Rather than functioning solely as a dashboard environment, the analyst workflow layer transforms the platform into an operational review system intended to support iterative investigation, contextual analysis, and structured operational awareness within a modular local SIEM and network defense architecture.



ZillaSIEM™ Analyst Workflow Pipeline

10. Networking & Exposure

The ZillaSIEM™ networking and exposure architecture was designed around a local-first operational model intended to preserve telemetry visibility while minimizing unnecessary service exposure and reducing operational attack surface. The networking model emphasizes trusted-LAN operation, bounded service accessibility, loopback-oriented internal components, and controlled telemetry ingest paths rather than broad internet-facing deployment assumptions.

The environment intentionally separates:

- Internal telemetry infrastructure
- Analyst-facing services
- Trusted telemetry sources
- External network boundaries

This separation improves operational clarity and supports a defense-oriented deployment posture suitable for local infrastructure monitoring and iterative engineering workflows.

At a high level, the networking model follows the principle:

Internal Processing Remains Local
Operational Access Remains Bounded
Public Exposure Remains Minimized

Networking Design Principles

Several operational principles guided the networking and exposure architecture.

Local-First Operation

The environment was designed primarily for operation within a trusted private network rather than as a publicly accessible cloud-hosted SIEM platform.

Telemetry processing, enrichment, evidence generation, and operational dashboards are intended to remain under local administrative control.

Minimized Attack Surface

Internal infrastructure components are intentionally shielded from unnecessary external exposure wherever practical.

Services such as:

- Postgres
- Loki
- Internal workers
- Evidence-processing pipelines

remain loopback-oriented or container-internal whenever possible.

Separation Of Internal & Operational Interfaces

The architecture distinguishes between:

- Internal runtime dependencies
- Telemetry ingest interfaces
- Dashboard access surfaces
- Administrative workflows

This separation improves operational containment and reduces coupling between internal telemetry systems and operator-facing services.

Bounded Operational Access

Operational interfaces are selectively exposed only to trusted LAN systems and administrative users.

The environment intentionally avoids unrestricted public exposure of dashboards, APIs, or telemetry infrastructure.

Internal Runtime Networking

Core runtime services communicate primarily through Docker networking and loopback-oriented host access patterns.

Internal runtime communication includes:

- Worker access to Loki
- API access to Postgres
- Grafana datasource connectivity
- Enrichment worker communication
- Structured evidence persistence
- Health-check workflows

Internal networking is intentionally isolated from external-facing operational interfaces wherever practical.

This architecture reduces exposure of telemetry internals while simplifying operational trust assumptions.

Loopback-Oriented Services

Several core services are intentionally configured for local-only accessibility.

Examples include:

- Postgres
- Loki
- Internal enrichment pipelines
- Worker coordination interfaces

Restricting these services to loopback or container-internal access provides several operational advantages:

- Reduced exposure risk

- Simplified firewall policy
- Reduced accidental access
- Improved operational containment
- Cleaner trust assumptions

The architecture intentionally avoids exposing raw telemetry stores directly to external clients.

Trusted-LAN Accessible Services

Some services are intentionally exposed to trusted LAN systems in order to support operational workflows and analyst review.

These services may include:

- Grafana dashboards
- ZillaAI interfaces
- ZillaSIEM™ API endpoints
- Operational status views

Access to these services is intentionally constrained through:

- Host firewall controls
- Limited port exposure
- Administrative network assumptions
- Token-authenticated workflows
- Local operational access policies

This model supports operational usability while avoiding unnecessary broad exposure.

Syslog Ingest Networking

Router and firewall telemetry enter the environment through the dedicated network syslog forwarder service.

The syslog pipeline assumes:

- Internal trusted telemetry sources
- Controlled infrastructure devices

- Trusted-LAN transport
- Known operational emitters

The syslog forwarder receives telemetry and normalizes labels before forwarding events into Loki.

Separating syslog ingest into a dedicated service improves modularity and operational clarity while isolating telemetry ingestion responsibilities from downstream processing and dashboard workflows.

API Ingest Workflows

Runtime metrics from endpoint systems are submitted through the ZillaSIEM™ API using token-authenticated workflows.

This ingest model intentionally avoids:

- Direct endpoint database access
- Unrestricted telemetry writes
- Broad internal service exposure

The API boundary acts as a controlled operational ingest interface supporting:

- Authentication validation
- Structured metric submission
- Health checks
- Operational summaries
- Dashboard integration

This architecture improves operational governance while reducing direct exposure of internal telemetry infrastructure.

Public Internet Boundary

The public internet is intentionally not treated as the primary operational access layer for the environment.

Direct exposure of:

- Loki
- Postgres

- Internal workers
- Structured evidence stores
- Enrichment services

is intentionally avoided.

This design reflects several operational priorities:

- Preserve local telemetry ownership
- Minimize external attack surface
- Reduce exposure complexity
- Maintain operator-controlled deployment
- Preserve infrastructure containment

Where remote access workflows are required, they are intended to occur through trusted administrative paths rather than unrestricted public-facing service exposure.

Docker Networking

Docker Compose orchestration provides service-level networking separation between runtime components.

Docker networking supports:

- Service discovery
- Internal container communication
- Runtime isolation
- Dependency management
- Simplified deployment workflows

This architecture also improves portability and reproducibility by allowing runtime relationships to remain codified within deployment configuration rather than manually managed host-level networking assumptions.

Operational Visibility & Exposure Monitoring

The environment incorporates operational validation and monitoring workflows intended to identify networking failures and telemetry degradation conditions.

Monitoring workflows may evaluate:

- Syslog continuity
- API responsiveness
- Dashboard accessibility
- Telemetry freshness
- Container runtime state
- Datasource health
- Worker execution continuity

This operational monitoring improves visibility into failures affecting the monitoring platform itself rather than only observing external traffic conditions.

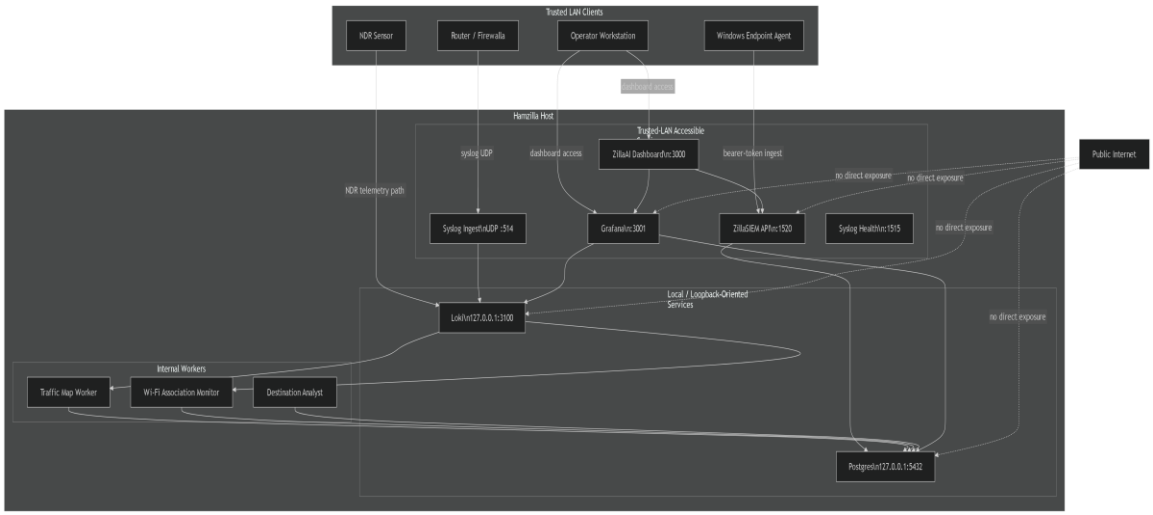
Exposure Philosophy

The networking model reflects a broader operational philosophy centered around:

- Controlled visibility
- Bounded access
- Local-first deployment
- Trusted-LAN operation
- Minimal external exposure
- Operational containment
- Maintainable infrastructure

Rather than attempting to maximize accessibility or cloud-native exposure, the environment prioritizes operational clarity, controlled deployment, and evidence-oriented telemetry visibility within a constrained and manageable trust boundary.

The resulting networking architecture therefore supports operational review, telemetry ingestion, enrichment workflows, and analyst access while intentionally minimizing unnecessary infrastructure exposure within the broader ZillaSIEM™ platform.



ZillaSIEM™ Networking & Exposure Model

11. Persistence & Retention

The ZillaSIEM™ persistence and retention architecture was designed to preserve operational continuity, maintain structured analytical evidence, and support long-term investigation workflows while balancing storage efficiency, query performance, and local deployment practicality. The platform intentionally separates raw telemetry retention from structured evidence persistence in order to support both operational visibility and analyst-oriented workflows without overloading dashboard systems with repeated high-volume event reconstruction.

The persistence model distinguishes between:

- Raw event telemetry
- Structured analytical evidence
- Runtime operational state
- Dashboard configuration
- Enrichment caches
- Historical attribution context

This layered approach improves maintainability, operational continuity, and workflow reuse across the broader SIEM and network defense environment.

At a high level, the persistence workflow follows the pattern:

Raw Telemetry → Event Retention →
Enrichment & Aggregation → Structured Evidence Persistence →
Operational Review & Historical Context

Persistence Design Principles

Several operational principles guided the persistence architecture.

Separate Raw Events From Structured Evidence

Raw telemetry streams remain in Loki, while enriched outputs, operational context, and analyst-oriented evidence are persisted into Postgres relational tables.

This separation improves dashboard responsiveness and reduces repeated expensive queries against large event datasets.

Preserve Investigation Continuity

Structured evidence should persist across dashboard refreshes, worker restarts, infrastructure maintenance, and operational review sessions.

Preserve Analyst Context

Attribution records, enrichment results, hypotheses, and operational metadata should remain available after raw telemetry ages out of short-term event storage.

Support Local Operational Scalability

Retention strategies should remain practical for local infrastructure deployment without requiring large-scale enterprise storage infrastructure.

Raw Telemetry Retention

Loki functions as the primary raw telemetry and event retention layer.

Retained event streams may include:

- Router syslog
- Firewall logs
- Suricata EVE events
- Infrastructure logs
- Runtime operational logs
- Telemetry freshness data
- Worker execution logs

The Loki retention model supports:

- Operational troubleshooting
- Dashboard queries
- Short-term investigations
- Stream validation
- Historical telemetry review

Because raw telemetry volume may grow rapidly, Loki retention is intentionally treated differently from structured evidence persistence.

The environment prioritizes preserving operationally meaningful aggregated evidence rather than indefinitely retaining all raw event streams.

Structured Evidence Persistence

Structured analytical outputs are persisted into Postgres relational tables.

Examples of structured evidence include:

- Traffic flow aggregates
- Threat hypotheses
- Destination enrichment
- Wi-Fi association history
- Runtime metrics
- Attribution records
- Reputation caches
- Operational summaries

This structured persistence model provides several operational benefits:

- Faster dashboard queries
- Reduced telemetry reconstruction overhead
- Improved historical continuity
- Stable analyst review workflows
- Persistent operational context

The platform intentionally stores evidence-oriented outputs rather than relying exclusively on ephemeral dashboard state or transient event searches.

Traffic Flow Retention

Traffic aggregation workers generate structured traffic summaries derived from Suricata telemetry and other enrichment workflows.

Persisted flow information may include:

- Source attribution
- Destination metadata
- Protocol observations
- Geographic context
- ASN ownership
- Traffic frequency
- Historical observation timing

Retaining aggregated traffic summaries improves operational review by preserving higher-level analytical context even after detailed raw telemetry ages out of event retention windows.

This model also supports historical trend analysis and operational comparison workflows.

Enrichment Cache Persistence

The platform maintains enrichment caches in order to reduce redundant external lookups and improve operational efficiency.

Persisted enrichment data may include:

- GeoIP metadata
- ASN ownership
- WHOIS/RDAP records
- Reverse DNS results
- Reputation observations
- Destination categorization

Caching enrichment results provides several operational advantages:

- Reduced lookup overhead
- Improved dashboard responsiveness
- Greater operational consistency
- Lower external dependency frequency
- Faster hypothesis generation

Enrichment persistence also improves repeatability across investigations by preserving historical context associated with previously observed destinations.

Wi-Fi Association History

Wi-Fi association records are persisted in order to support attribution continuity and infrastructure visibility.

Persisted data may include:

- Device association timing
- SSID history
- MAC address observations
- DHCP relationships
- Local IP history
- Association churn

This retention model assists with:

- NAT disambiguation
- Endpoint attribution
- Device history analysis
- Infrastructure troubleshooting
- Operational investigation continuity

The platform intentionally preserves historical association context rather than relying exclusively on transient current-state observations.

Threat Hypothesis Persistence

Generated threat hypotheses are persisted into structured relational tables along with:

- Severity
- Confidence
- Supporting evidence
- Analyst context
- Disposition guidance
- Operational notes

Persisting hypotheses supports:

- Workflow continuity
- Investigation review
- Historical comparison
- Analyst documentation
- Operational reporting
- Future workflow automation

The persistence architecture intentionally treats analytical outputs as operational evidence rather than transient dashboard artifacts.

Runtime Metrics Retention

Runtime metrics are retained to support infrastructure monitoring, operational validation, and health analysis workflows.

Persisted runtime metrics may include:

- Host operational state
- Container health
- Resource utilization
- Service responsiveness
- Telemetry freshness observations
- Endpoint status information

This historical runtime context improves troubleshooting visibility and supports post-incident operational review.

Dashboard & Provisioning Persistence

Grafana runtime state is persisted separately from repo-managed provisioning assets.

Persisted dashboard state may include:

- Dashboard configuration
- Datasource configuration
- Alert definitions
- User operational preferences
- Runtime dashboard metadata

At the same time, dashboards, datasources, and alerting logic are also maintained within repository-managed provisioning files to improve:

- Deployment reproducibility
- Configuration consistency
- Version control
- Infrastructure portability
- Operational recovery workflows

This dual-layer persistence model balances operational flexibility with deployment repeatability.

Retention Lifecycle Philosophy

The retention strategy reflects a broader operational philosophy focused on:

- Preserving useful evidence
- Avoiding unnecessary telemetry sprawl
- Supporting operational continuity
- Maintaining local deployment practicality
- Prioritizing structured analytical value

The platform intentionally distinguishes between:

- Short-lived raw telemetry
- Long-lived analytical evidence

- Runtime operational state
- Persistent investigation context

This layered retention model improves operational usability while reducing storage pressure and query overhead associated with indefinite raw telemetry retention.

Operational Benefits

The persistence architecture supports several broader operational objectives:

- Stable dashboard performance
- Historical investigation continuity
- Evidence-oriented workflows
- Attribution preservation
- Enrichment reuse
- Operational troubleshooting
- Detection tuning
- Infrastructure validation

Rather than functioning as a temporary event viewer, the platform preserves structured operational knowledge intended to support iterative investigation, infrastructure visibility, and long-term operational awareness within the broader ZillaSIEM™ architecture.

ZillaSIEM™ Persistence & Retention Model

12. Operational Validation

The ZillaSIEM™ operational validation and health monitoring architecture was designed to ensure runtime stability, telemetry continuity, infrastructure visibility, and reliable analyst workflows across the broader SIEM and network defense environment. In addition to monitoring external telemetry and network activity, the platform intentionally monitors the health and integrity of the monitoring environment itself.

This operational philosophy recognizes that visibility failures, telemetry degradation, datasource instability, and infrastructure faults can significantly impact the reliability of detection workflows and analyst decision-making. As a result, the platform treats operational validation as a core architectural function rather than an auxiliary administrative task.

The operational validation model focuses on:

- Telemetry continuity
- Runtime stability
- Service availability
- Datasource integrity
- Dashboard consistency
- Worker execution health
- Post-deployment verification
- Infrastructure observability

At a high level, the operational validation workflow follows the sequence:

Service Health → Telemetry Validation →
Runtime Monitoring → Alert Generation →
Operational Review → Corrective Action

Operational Validation Philosophy

Several operational principles guided the design of the validation and health monitoring architecture.

Monitor The Monitoring System

The platform intentionally validates the integrity of telemetry ingestion, dashboards, workers, and infrastructure dependencies rather than assuming monitoring systems are always functioning correctly.

Preserve Telemetry Continuity

Operational visibility depends on reliable telemetry flow. Missing telemetry streams are treated as operationally significant conditions rather than silent failures.

Validate Runtime Dependencies

Critical dependencies such as databases, APIs, dashboards, and workers are continuously evaluated to improve reliability and troubleshooting visibility.

Preserve Human Visibility

Operational health data is surfaced through dashboards and alerts in order to support rapid troubleshooting and infrastructure review.

Service Health Monitoring

The environment continuously evaluates the health of core runtime services including:

- ZillaSIEM™ API
- Grafana
- Loki
- Postgres
- Syslog forwarders
- Enrichment workers
- Analysis pipelines
- Runtime collectors

Service validation workflows may evaluate:

- Process availability
- Container state
- API responsiveness
- Datasource connectivity
- Worker execution cadence
- Service restart behavior

Operational health observations are surfaced through dashboards, alerts, and operational summaries.

Telemetry Freshness Monitoring

Telemetry freshness monitoring forms a major component of the operational validation architecture.

Freshness validation evaluates whether telemetry streams continue updating within expected operational intervals.

Monitored telemetry categories may include:

- Router syslog
- Firewall telemetry
- Suricata event streams
- Runtime metrics
- Worker-generated evidence
- Dashboard datasource responses

The platform intentionally treats stale telemetry as an operational detection category because degraded visibility may indicate:

- Ingestion failures
- Worker crashes
- Network interruptions
- Infrastructure instability
- Service outages
- Runtime misconfiguration

This approach improves operational awareness by detecting failures affecting the monitoring environment itself.

Runtime Metrics & Infrastructure Visibility

The environment collects and persists runtime metrics in order to improve visibility into host and service behavior.

Observed runtime information may include:

- Host operational state
- Resource utilization
- Service responsiveness

- Worker execution timing
- Container runtime health
- Telemetry latency
- Operational drift indicators

Persisting runtime metrics supports:

- Historical troubleshooting
- Performance analysis
- Infrastructure validation
- Operational review workflows
- Post-incident analysis

The platform intentionally combines infrastructure visibility with telemetry analysis rather than treating them as unrelated operational domains.

Dashboard Validation

Grafana dashboards form the primary operational review surface for the environment. As a result, dashboard consistency and datasource integrity are treated as important operational concerns.

Validation workflows may evaluate:

- Datasource availability
- Dashboard rendering consistency
- Alert rule status
- Query responsiveness
- Provisioning integrity
- Panel update cadence

Operational validation also includes ensuring that dashboards continue presenting meaningful telemetry after:

- Service restarts
- Infrastructure maintenance
- Configuration updates

- Runtime redeployments
- Worker changes

This improves reliability and reduces operational ambiguity during troubleshooting workflows.

Query Validation & Worker Monitoring

Processing workers continuously generate structured evidence used by dashboards and analyst workflows. Validation mechanisms monitor these workers in order to identify:

- Query failures
- Stalled processing
- Missing evidence generation
- Enrichment failures
- Correlation breakdowns
- Worker instability

The environment intentionally monitors both infrastructure state and evidence-generation behavior because silent enrichment failures may reduce analytical visibility even while infrastructure services appear healthy.

Operational validation therefore extends beyond container uptime into verification of functional evidence generation.

Docker Runtime Validation

Docker Compose orchestration simplifies deployment but introduces operational dependencies between containers, networking, storage, and service ordering.

Validation workflows may evaluate:

- Container state
- Restart conditions
- Dependency availability
- Service health ordering
- Volume accessibility
- Runtime networking

Operational monitoring helps identify scenarios in which:

- Services restart unexpectedly
- Dependencies fail to initialize
- Networking assumptions change
- Volumes become unavailable
- Telemetry pipelines stop updating

This improves deployment reliability and troubleshooting visibility.

Post-Reboot Validation

The environment incorporates post-reboot operational validation workflows intended to verify that telemetry and services resume correctly after host restarts or maintenance activities.

Post-reboot validation may include:

- API health checks
- Grafana accessibility
- Loki query testing
- Telemetry freshness verification
- Worker execution validation
- Dashboard consistency review
- Container status inspection

This workflow reduces the risk of silent telemetry outages or partially degraded operational states after infrastructure changes.

Alerting & Operational Visibility

Operational health conditions are surfaced through Grafana dashboards and alerting workflows.

Health-related alerts may include:

- Missing telemetry streams
- Worker execution failures
- Datasource degradation

- Container instability
- Infrastructure drift
- Runtime anomalies
- Dashboard provisioning failures

The platform intentionally surfaces operational degradation alongside security-oriented telemetry in order to preserve confidence in the reliability of the monitoring environment itself.

Human Operational Review

Human review remains an intentional part of operational validation workflows.

Operators may:

- Validate telemetry continuity
- Confirm infrastructure stability
- Review alert accuracy
- Investigate degraded services
- Tune health thresholds
- Validate dashboard behavior
- Confirm evidence consistency

This governance model reflects the broader architectural philosophy of preserving human oversight and evidence-oriented operational workflows.

Operational Outcomes

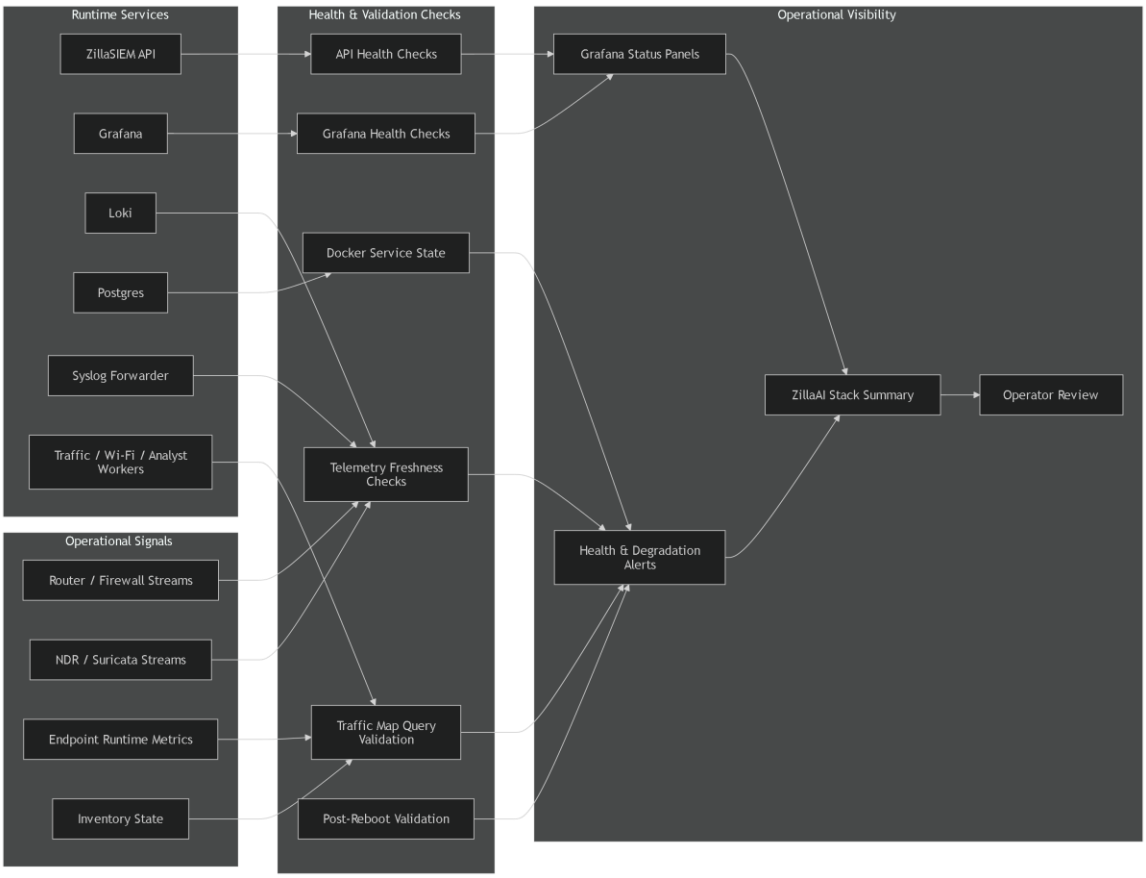
Operational validation workflows may ultimately result in:

- Infrastructure remediation
- Service restarts
- Dashboard correction
- Query tuning
- Worker troubleshooting
- Detection refinement

- Deployment adjustments
- Evidence preservation

The operational validation architecture therefore transforms the environment from a passive telemetry viewer into a continuously monitored operational platform intended to maintain reliable visibility, stable evidence generation, and trustworthy analyst workflows across the broader ZillaSIEM™ deployment.

Rather than assuming infrastructure stability by default, the platform continuously validates the integrity of the telemetry and operational systems upon which all downstream detection and analysis workflows depend.



ZillaSIEM™ Operational Validation & Health Monitoring

13. AI-Assisted Engineering Workflow

The ZillaSIEM™ platform was developed through an iterative engineering workflow that combined traditional infrastructure engineering practices with bounded AI-assisted implementation support. The workflow was intentionally designed to preserve human operational control, deployment governance, and runtime validation while leveraging AI

assistance to accelerate implementation, troubleshooting, documentation, and operational iteration.

Rather than treating AI as an autonomous deployment system, the environment adopted a constrained operational model in which AI-assisted workflows functioned as engineering support mechanisms operating under explicit operator review and approval.

This engineering methodology reflects a broader operational philosophy centered around:

- Human-governed infrastructure management
- Incremental implementation
- Evidence-oriented troubleshooting
- Controlled deployment workflows
- Operational validation
- Bounded AI assistance

At a high level, the workflow followed the sequence:

Operational Goal → AI-Assisted Inspection & Generation →
Human Review → Controlled Deployment →
Runtime Validation → Iterative Refinement

Engineering Workflow Objectives

Several operational objectives influenced the AI-assisted engineering methodology.

Accelerate Iteration

AI assistance was used to reduce repetitive implementation effort and accelerate operational iteration while preserving deployment oversight.

Preserve Human Governance

Operators retained authority over infrastructure decisions, deployment approval, exposure configuration, and runtime validation.

Improve Documentation & Consistency

AI assistance supported generation of deployment documentation, configuration structure, operational summaries, and troubleshooting narratives.

Maintain Operational Visibility

Infrastructure changes remained observable and reviewable through runtime validation, health monitoring, logs, and dashboards.

Role Of Codex

Codex was used as an engineering assistant throughout the development and operational refinement of the platform.

AI-assisted activities included:

- Repository inspection
- Configuration generation
- Docker Compose updates
- Service integration assistance
- Dashboard provisioning support
- Query refinement
- Troubleshooting guidance
- Documentation generation
- Operational workflow analysis

The workflow intentionally treated generated outputs as proposals requiring review rather than automatically trusted infrastructure changes.

Operators reviewed:

- Configuration changes
- Runtime assumptions
- Service exposure
- Deployment logic
- Network boundaries
- Persistence configuration
- Operational impacts

before approving implementation or deployment actions.

Repository Inspection & Analysis

AI-assisted workflows were used to analyze:

- Existing repository structure

- Runtime service relationships
- Deployment configuration
- Dashboard provisioning assets
- Worker interactions
- API integration boundaries
- Operational dependencies

This accelerated understanding of service relationships and reduced manual effort associated with navigating evolving infrastructure configurations.

Repository inspection also supported identification of:

- Configuration inconsistencies
- Provisioning errors
- Runtime dependency issues
- Deployment drift
- Integration gaps

during iterative development workflows.

Configuration & Infrastructure Generation

AI assistance supported generation and refinement of infrastructure artifacts including:

- Docker Compose definitions
- Service configuration
- Grafana provisioning files
- Datasource configuration
- Alert rule structure
- API integration logic
- Operational scripts
- Documentation templates

Generated outputs were subsequently validated through deployment review and runtime testing workflows.

The platform intentionally preserved explicit operator review before applying changes affecting:

- Service exposure
- Runtime persistence
- Container orchestration
- Networking assumptions
- Telemetry pipelines
- Operational dependencies

This governance model reduced the likelihood of unsafe or unreviewed infrastructure modification.

Controlled Deployment Workflow

Deployment activities remained human-controlled and operationally bounded throughout the engineering process.

Deployment workflows included:

- Configuration review
- Compose validation
- Runtime inspection
- Controlled execution
- Post-deployment verification
- Health validation
- Dashboard review
- Telemetry continuity checks

Operational execution occurred through controlled remote management workflows rather than unrestricted automated deployment pipelines.

This model preserved operational accountability while still benefiting from accelerated implementation assistance.

Runtime Validation & Verification

A major operational principle within the workflow was validating runtime behavior after deployment or infrastructure modification.

Validation activities included:

- API health checks
- Grafana accessibility testing
- Loki query validation
- Telemetry freshness review
- Dashboard inspection
- Worker execution confirmation
- Structured evidence verification
- Container state review

This verification model ensured that infrastructure changes were evaluated based on actual operational behavior rather than configuration assumptions alone.

The workflow intentionally prioritized observable operational outcomes over theoretical deployment correctness.

Troubleshooting & Iterative Refinement

AI-assisted workflows were particularly useful during troubleshooting and iterative operational refinement activities.

Examples included:

- Dashboard provisioning correction
- Container dependency troubleshooting
- Loki query refinement
- Datasource validation
- Runtime health analysis
- Telemetry debugging
- Worker behavior inspection
- Configuration restructuring

Rather than attempting fully automated remediation, AI assistance accelerated investigation and proposal generation while preserving human review over corrective actions.

This workflow improved iteration speed without removing operational accountability.

Human Approval & Governance

Human approval remained an explicit requirement for high-impact operational activities.

Examples included:

- Infrastructure exposure changes
- Host reboots
- Service restarts
- Firewall modifications
- Runtime deployments
- Persistent storage changes
- Dependency restructuring

This governance model preserved operational safety and accountability while ensuring that AI assistance remained bounded within reviewable workflows.

The environment intentionally avoided autonomous infrastructure modification without operator validation.

AI-Assisted Documentation

AI assistance also supported operational documentation generation and refinement throughout the project lifecycle.

Documentation support included:

- Architecture summaries
- Workflow explanations
- Operational narratives
- Deployment structure
- Diagram development
- Technical whitepaper drafting
- Troubleshooting documentation

- Runtime analysis summaries

This improved documentation consistency and reduced the overhead associated with maintaining large volumes of engineering and operational reference material.

Operational Benefits

The AI-assisted engineering workflow provided several practical operational benefits:

- Faster infrastructure iteration
- Reduced repetitive configuration effort
- Improved troubleshooting velocity
- Better documentation consistency
- Accelerated deployment refinement
- Improved visibility into runtime relationships
- Faster operational experimentation

At the same time, the environment preserved:

- Human validation
- Deployment governance
- Runtime verification
- Infrastructure accountability
- Operational oversight

throughout the engineering lifecycle.

Engineering Philosophy

The engineering workflow reflects a broader philosophy that AI assistance should augment infrastructure engineering rather than replace operational judgment or deployment accountability.

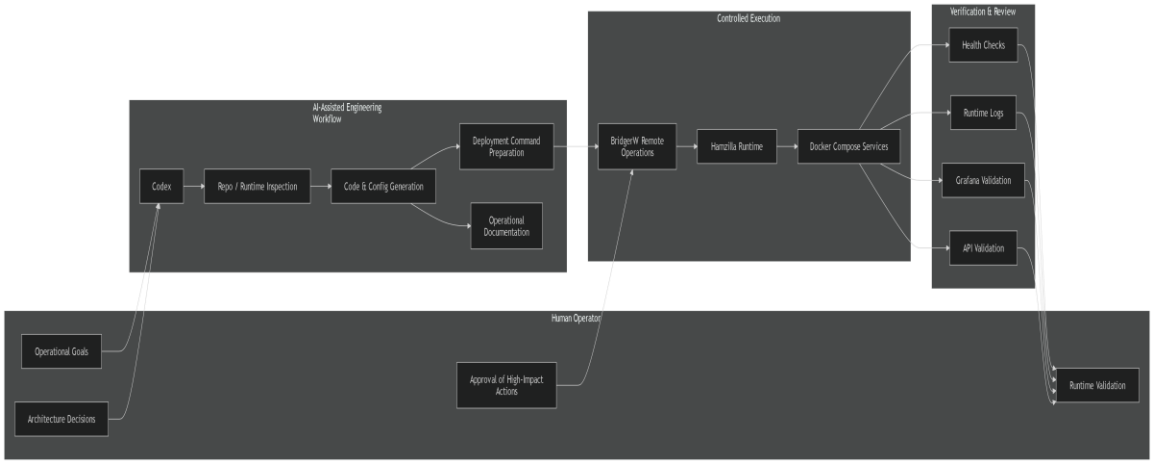
The platform therefore demonstrates a practical example of:

- AI-assisted infrastructure engineering
- Human-governed deployment
- Evidence-oriented validation

- Operationally bounded automation
- Iterative local-first engineering

within a modular SIEM and network defense environment.

Rather than presenting AI as an autonomous security operator, the workflow illustrates how AI-assisted tooling can support infrastructure development, operational troubleshooting, and documentation workflows while preserving human control over deployment, exposure, and runtime validation decisions.



AI-Assisted Engineering & Validation Workflow

14. Incident Investigation Workflow

The ZillaSIEM™ incident investigation workflow was designed to support evidence-oriented operational review, contextual analysis, attribution validation, and structured investigative continuity within a local SIEM and network defense environment. Rather than relying exclusively on isolated alerts or static dashboards, the workflow integrates telemetry review, enrichment context, historical evidence, operational health information, and analyst validation into a layered investigative process.

The workflow architecture emphasizes:

- Incremental evidence review
- Context preservation
- Attribution transparency
- Investigation continuity
- Human-guided analysis

- Structured operational visibility

The environment intentionally treats investigations as iterative analytical workflows rather than fully automated decision pipelines. Alerts, anomalies, and hypotheses serve as operational starting points that require contextual review and evidence validation before escalation or remediation decisions are made.

At a high level, the investigation workflow follows the sequence:

Operational Signal → Evidence Collection →
Correlation & Enrichment → Attribution Review →
Threat Hypothesis → Analyst Validation →
Operational Outcome

Investigation Workflow Philosophy

Several operational principles guided the design of the incident investigation workflow.

Preserve Supporting Evidence

Analysts should retain access to the telemetry and enrichment data supporting a detection or operational hypothesis.

Preserve Attribution Context

Source attribution, Wi-Fi association history, inventory relationships, and runtime state should remain visible throughout the investigation lifecycle.

Preserve Analytical Uncertainty

The workflow intentionally avoids presenting incomplete observations as definitive conclusions where evidence remains ambiguous or unresolved.

Support Incremental Review

Investigations should support movement between high-level summaries and detailed operational evidence without losing contextual continuity.

Maintain Human Oversight

AI-assisted workflows may assist operational review, summarization, and investigation guidance, but human analysts remain responsible for operational interpretation and decision-making.

Investigation Entry Points

Investigations may begin from several operational triggers within the platform.

Examples include:

- IDS signature alerts
- Traffic anomalies
- Suspicious destination observations
- Runtime instability
- Telemetry freshness degradation
- Infrastructure health alerts
- Threat hypotheses
- Dashboard anomalies
- Attribution inconsistencies

These signals function as investigation initiators rather than authoritative indicators of malicious activity.

The architecture intentionally preserves operational visibility into uncertain or unresolved conditions.

Initial Evidence Collection

The first stage of investigation focuses on collecting supporting operational evidence associated with the triggering observation.

Evidence collection may include:

- Raw Loki event review
- Suricata event inspection
- Traffic flow analysis
- Runtime metric validation
- Wi-Fi association history
- Asset inventory context
- Dashboard drilldowns
- Historical observation review

The workflow attempts to consolidate evidence from multiple telemetry sources into a unified operational context for analyst review.

This reduces fragmentation between infrastructure telemetry, network observations, and runtime visibility.

Correlation & Attribution Review

A major component of the investigation workflow involves validating source attribution and contextual relationships associated with observed activity.

Correlation workflows may evaluate:

- Source IP relationships
- NAT observations
- Wi-Fi association state
- Asset identity
- Inventory relationships
- Runtime metrics
- Historical traffic patterns
- Device activity timing

The environment intentionally preserves attribution uncertainty where evidence remains incomplete.

This helps distinguish between:

- Confirmed attribution
- Probable attribution
- Shared infrastructure activity
- Ambiguous observations
- Unknown sources

Preserving attribution transparency improves analyst confidence and reduces the risk of unsupported assumptions during operational review.

Destination Analysis & Enrichment

Observed destinations are evaluated using multiple enrichment workflows intended to improve operational understanding.

Destination analysis may include:

- GeoIP information
- ASN ownership
- WHOIS/RDAP context
- Reverse DNS analysis
- Reputation observations
- Historical traffic frequency
- Protocol usage patterns
- Geographic relationships

The platform intentionally separates enrichment from raw telemetry retention in order to preserve operational continuity and reduce redundant lookup activity.

Enrichment outputs provide contextual support for analyst interpretation but do not independently determine maliciousness.

Threat Hypothesis Evaluation

The destination analyst worker generates operational hypotheses based on observed telemetry and enrichment context.

Analyst review of hypotheses may consider:

- Severity
- Confidence
- Attribution quality
- Infrastructure context
- Reputation consistency
- Historical observations
- Protocol behavior
- Geographic anomalies
- Telemetry completeness

The workflow intentionally uses hypothesis-oriented language to preserve analytical caution and operational transparency.

Human analysts remain responsible for validating or dismissing investigative conclusions.

AI-Assisted Investigation Support

ZillaAI consumes structured evidence and operational summaries from ZillaSIEM™ in order to support AI-assisted investigative workflows.

AI-assisted investigation support may include:

- Operational summarization
- Context aggregation
- Investigation guidance
- Threat explanation
- Workflow continuity
- Evidence organization
- Dashboard interpretation

However, the architecture intentionally preserves analyst access to underlying telemetry and operational evidence rather than relying exclusively on AI-generated summaries.

The workflow therefore treats AI assistance as an analytical support layer rather than an autonomous investigation authority.

Operational Drilldowns

Investigations may move iteratively between high-level operational summaries and lower-level telemetry review.

Drilldown workflows may include transitions between:

- Dashboard panels
- Traffic maps
- Threat hypotheses
- Raw event streams
- Runtime metrics
- Wi-Fi association views
- Destination summaries
- Asset inventory context

This layered review model improves investigation continuity and reduces operational fragmentation between telemetry systems.

Investigation Outcomes

Investigations may ultimately result in several operational outcomes.

Examples include:

- Continued monitoring
- Detection tuning
- Infrastructure remediation
- Dashboard refinement
- Evidence documentation
- Operational escalation
- Attribution updates
- Telemetry troubleshooting
- Workflow refinement

The architecture intentionally treats investigations as operational learning workflows capable of improving future telemetry visibility and analytical accuracy.

Evidence Preservation

Structured evidence associated with investigations is persisted into Postgres relational tables in order to support:

- Historical continuity
- Analyst review
- Workflow reuse
- Dashboard integration
- Future automation
- Operational documentation

Persisted evidence may include:

- Threat hypotheses

- Attribution records
- Enrichment context
- Runtime observations
- Investigation summaries
- Operational notes

This persistence model improves continuity across sessions and preserves operational context after raw telemetry ages out of short-term retention windows.

Human Validation & Operational Governance

Human oversight remains central to the investigation architecture.

Operators and analysts may:

- Validate evidence quality
- Review attribution assumptions
- Assess confidence levels
- Investigate ambiguous activity
- Tune workflows
- Refine hypotheses
- Document findings
- Determine operational significance

This governance model reflects the broader architectural philosophy of evidence-oriented analysis, bounded AI assistance, and operator-controlled review.

Operational Value

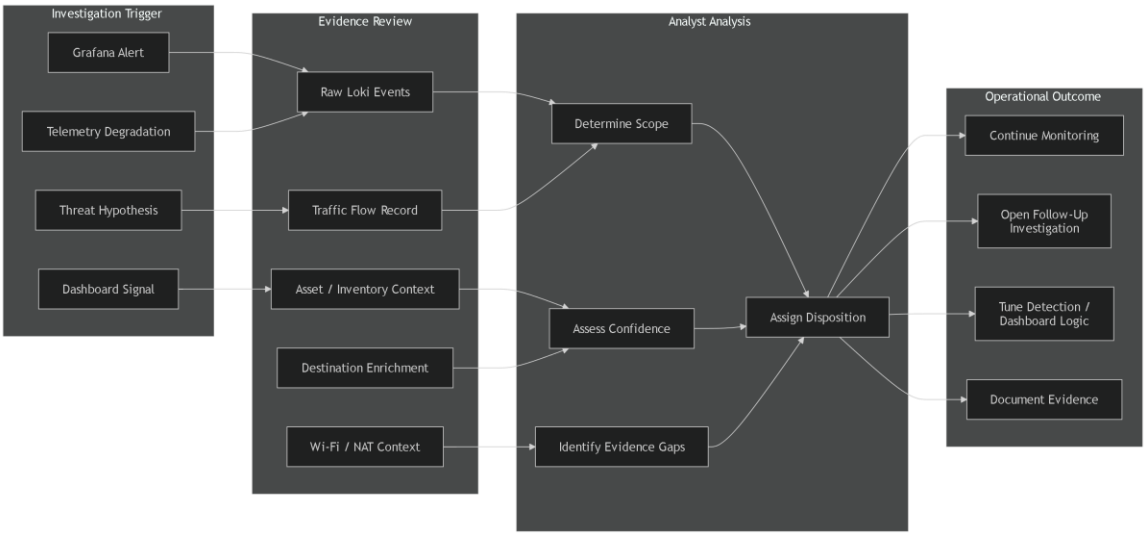
The incident investigation workflow transforms the platform from a passive telemetry aggregation environment into an operational review and investigative analysis system intended to support:

- Contextual visibility
- Structured investigations
- Attribution analysis
- Evidence continuity

- Human-guided review
- Operational awareness

within the broader ZillaSIEM™ SIEM and network defense architecture.

Rather than functioning solely as an alert-generation environment, the workflow supports iterative operational analysis and evidence-driven investigation designed to improve visibility, maintain continuity, and preserve analytical transparency throughout the investigation lifecycle.



ZillaSIEM™ Incident Investigation Workflow

15. Repository Ownership Model

The ZillaSIEM™ platform was intentionally designed with a separation between telemetry infrastructure responsibilities and higher-level operator workflow responsibilities. This architectural separation resulted in two primary repository domains:

- ZillaSIEM™
- ZillaAI

The repository ownership model was created to improve maintainability, operational clarity, modularity, and long-term extensibility while reducing unnecessary coupling between telemetry processing systems and operator-facing workflow surfaces.

Rather than consolidating all functionality into a single monolithic repository, the environment separates responsibilities according to operational role and infrastructure ownership.

At a high level, the ownership model follows the principle:

ZillaSIEM™ Owns Telemetry & Evidence
ZillaAI Owns Operator Workflow & Interaction

This separation allows each environment to evolve independently while still supporting integrated operational workflows through bounded APIs, dashboards, and shared operational context.

Architectural Separation Philosophy

Several operational principles guided the repository ownership architecture.

Preserve Separation Of Concerns

Telemetry ingestion, enrichment, evidence generation, and dashboard infrastructure remain logically separate from operator-facing workflow orchestration and AI-assisted interaction layers.

Reduce Coupling

Changes to telemetry infrastructure should not require unnecessary modification of analyst workflow logic, and vice versa.

Improve Maintainability

Independent repositories improve operational organization and simplify long-term engineering workflows.

Support Modular Expansion

Additional telemetry processors, dashboards, AI workflows, or orchestration features can evolve independently without restructuring the entire operational environment.

ZillaSIEM™ Responsibilities

The ZillaSIEM™ repository owns the telemetry and operational infrastructure layer of the environment.

Primary responsibilities include:

- Telemetry collection
- Syslog ingestion
- Loki integration
- Structured evidence persistence
- Traffic analysis

- Threat enrichment
- Detection workflows
- Dashboard provisioning
- Alerting logic
- Runtime health monitoring
- Operational APIs
- Worker orchestration

ZillaSIEM™ therefore acts as the operational telemetry and evidence-generation platform underlying the broader environment.

Telemetry Ownership

The repository contains the logic and infrastructure responsible for:

- Router telemetry ingestion
- Firewall telemetry collection
- Suricata event handling
- Runtime metric ingest
- Wi-Fi association analysis
- Traffic aggregation
- Destination enrichment
- Structured evidence persistence

This separation ensures that telemetry processing remains centralized and operationally consistent.

Infrastructure Ownership

ZillaSIEM™ also owns deployment infrastructure including:

- Docker Compose orchestration
- Container definitions
- Provisioning assets
- Datasource configuration

- Alert definitions
- Worker configuration
- Runtime validation logic

Maintaining infrastructure assets within the SIEM repository improves deployment reproducibility and operational consistency.

Dashboard & Alerting Ownership

Grafana provisioning, dashboard definitions, alerting logic, and datasource configuration are managed within the ZillaSIEM™ repository.

This ownership model ensures that dashboards remain tightly aligned with:

- Telemetry structure
- Evidence tables
- Detection logic
- Runtime services
- Operational validation workflows

rather than being externally dependent on loosely managed visualization assets.

ZillaAI Responsibilities

The ZillaAI repository owns the operator interaction and AI-assisted workflow layer of the environment.

Primary responsibilities include:

- Operator dashboards
- AI-assisted summaries
- Analyst workflow orchestration
- Context aggregation
- Operational narrative generation
- Investigation support
- Workflow continuity
- Human interaction surfaces

ZillaAI consumes outputs generated by ZillaSIEM™ rather than directly owning telemetry ingestion or evidence-generation infrastructure.

Operational Workflow Ownership

ZillaAI focuses on improving the usability and operational interpretation of SIEM outputs.

Workflow responsibilities may include:

- Alert summarization
- Threat explanation
- Operational guidance
- Investigation continuity
- Contextual review surfaces
- Analyst-oriented summaries
- Workflow coordination

This separation allows AI-assisted workflows to evolve independently from telemetry infrastructure.

Bounded Integration Model

ZillaAI interacts with ZillaSIEM™ through:

- APIs
- Dashboard integrations
- Structured evidence consumption
- Operational summaries
- Runtime health endpoints

The environment intentionally avoids tightly coupling ZillaAI directly to raw telemetry stores or internal infrastructure components.

This bounded integration model improves maintainability and reduces architectural complexity.

Shared Operational Context

Although responsibilities are separated, both repositories operate within a shared operational ecosystem.

Shared operational context may include:

- Threat hypotheses
- Runtime health information
- Dashboard state
- Operational summaries
- Detection results
- Structured evidence
- Attribution context

The repositories therefore cooperate operationally while maintaining distinct ownership boundaries.

Operational Benefits Of Repository Separation

Separating the repositories provides several practical operational advantages.

Improved Maintainability

Telemetry infrastructure and AI-assisted workflow logic can evolve independently.

Cleaner Deployment Boundaries

Infrastructure deployments remain separate from operator workflow updates.

Reduced Risk During Iteration

Changes to dashboards or AI workflows are less likely to unintentionally impact telemetry ingestion or structured evidence persistence.

Easier Troubleshooting

Operational failures can be isolated more clearly between telemetry infrastructure and analyst-facing systems.

Better Long-Term Extensibility

Additional workers, dashboards, enrichment pipelines, or AI-assisted workflows can be added incrementally without requiring large-scale architectural restructuring.

Governance & Operational Ownership

The repository ownership model also supports clearer operational governance.

Examples include:

- Telemetry infrastructure ownership
- Dashboard provisioning ownership
- Runtime validation ownership
- Workflow orchestration ownership
- AI-assisted interaction ownership
- Structured evidence ownership

This clarity improves operational discipline and reduces ambiguity regarding infrastructure responsibility and deployment management.

Integration Through APIs & Structured Evidence

The repositories intentionally integrate through bounded operational interfaces rather than direct shared runtime ownership.

Examples include:

- Structured evidence APIs
- Dashboard datasource integrations
- Runtime health endpoints
- Operational summaries
- Evidence retrieval workflows

This architecture improves flexibility while reducing hidden dependencies between telemetry systems and workflow orchestration layers.

Operational Philosophy

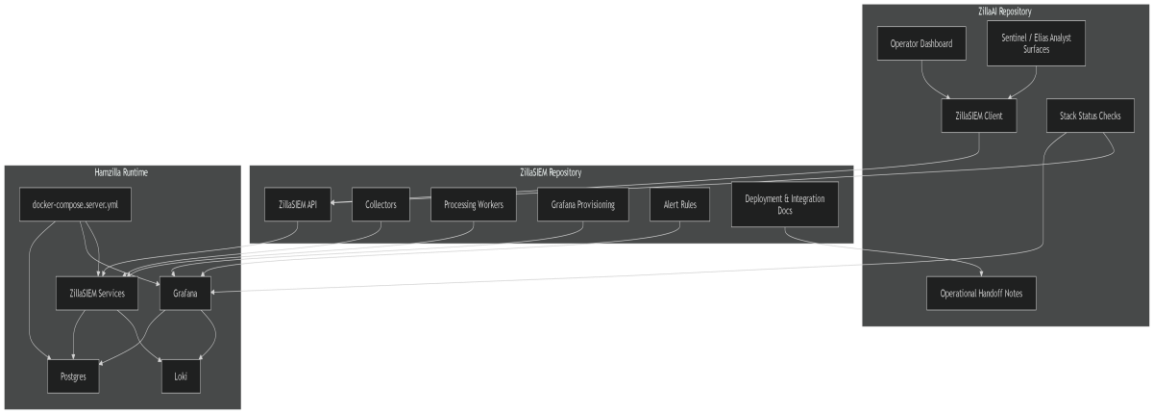
The repository ownership model reflects a broader operational philosophy centered around:

- Modular engineering
- Separation of concerns
- Maintainable infrastructure
- Bounded integrations
- Independent iteration
- Evidence-oriented workflows

- Operator-controlled deployment

Rather than consolidating all functionality into a single operational domain, the architecture intentionally separates telemetry infrastructure from operator workflow orchestration in order to preserve clarity, maintainability, and long-term extensibility.

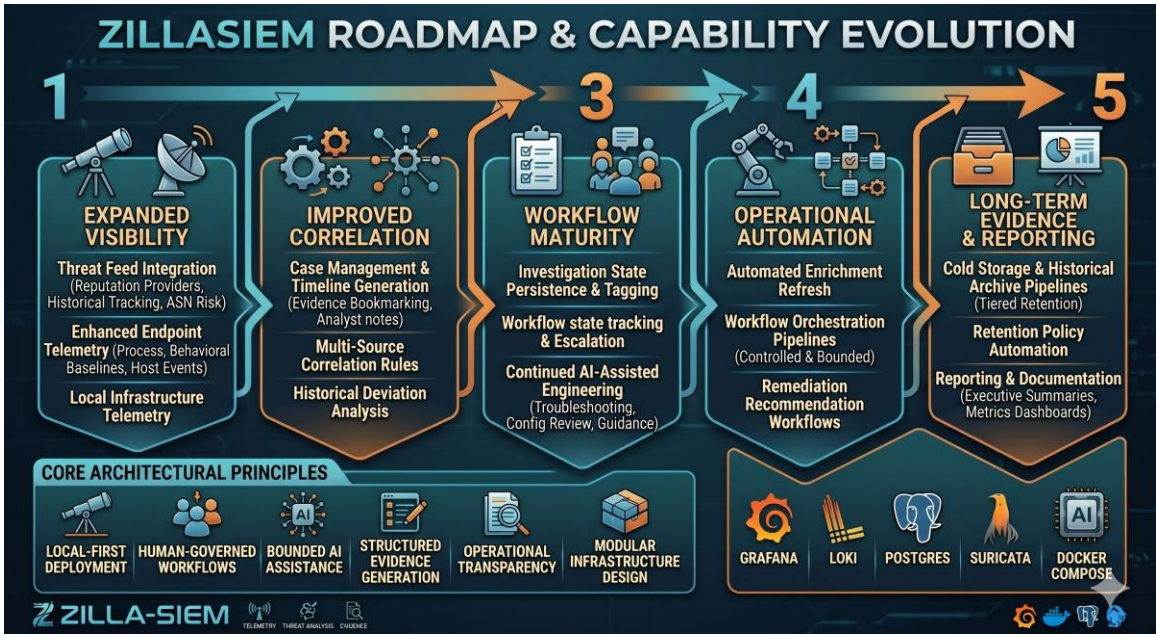
The resulting model supports iterative engineering, AI-assisted operational workflows, structured evidence generation, and maintainable local SIEM infrastructure within the broader ZillaSIEM™ operational ecosystem.



ZillaAI / ZillaSIEM™ Repository Ownership Model

16. Roadmap & Future Enhancements

The ZillaSIEM™ platform was intentionally designed as a modular and iterative operational environment capable of evolving over time as telemetry requirements, analytical workflows, operational visibility goals, and infrastructure capabilities expand. Rather than treating the environment as a static deployment, the architecture supports incremental enhancement through independently maintainable services, bounded integrations, structured evidence persistence, and modular workflow composition.



The roadmap for the platform focuses on extending operational visibility, improving analytical workflows, strengthening evidence continuity, and expanding automation capabilities while preserving the project's core architectural principles:

- Local-first deployment
- Human-governed workflows
- Bounded AI assistance
- Structured evidence generation
- Operational transparency
- Modular infrastructure design

The roadmap therefore emphasizes operational maturity and maintainability rather than rapid feature expansion without architectural discipline.

At a high level, future enhancement goals follow the progression:

Expanded Visibility → Improved Correlation →
Workflow Maturity → Operational Automation →
Long-Term Evidence & Reporting

Roadmap Philosophy

Several operational principles guide future platform development.

Preserve Modular Architecture

Future enhancements should integrate through bounded services, APIs, or evidence workflows rather than tightly coupling new functionality directly into core telemetry infrastructure.

Preserve Human Oversight

AI-assisted workflows and automation should continue operating under operator review and bounded execution assumptions.

Improve Operational Continuity

Enhancements should improve evidence retention, investigation continuity, operational visibility, and workflow consistency.

Expand Analytical Context

Future development should improve contextual visibility, attribution quality, and evidence correlation without sacrificing operational transparency.

Expanded Threat Intelligence Integration

One planned area of future enhancement involves expanding external threat intelligence and contextual enrichment capabilities.

Potential enhancements may include:

- Additional reputation providers
- Threat feed integration
- IOC enrichment
- Domain categorization
- Historical reputation tracking
- ASN risk analysis
- Threat scoring workflows
- Intelligence correlation caching

The platform intentionally separates enrichment pipelines from core telemetry ingestion, which allows new intelligence workflows to be introduced incrementally without restructuring the broader architecture.

Future threat intelligence workflows are expected to prioritize contextual analyst support rather than opaque automated classification.

Case Management & Investigation Continuity

Future workflow improvements may include structured case management capabilities intended to improve investigation continuity and operational documentation.

Potential enhancements may include:

- Investigation tracking
- Evidence bookmarking
- Analyst case notes
- Timeline generation
- Workflow state persistence
- Investigation tagging
- Escalation tracking
- Evidence export workflows

These capabilities would extend the platform's evidence-oriented operational model while improving continuity between investigations and operational review sessions.

Expanded Endpoint Visibility

The current runtime metrics pipeline provides foundational endpoint operational visibility. Future enhancements may expand endpoint telemetry capabilities to include additional:

- Process visibility
- Host event collection
- Service state monitoring
- Runtime anomaly detection
- Endpoint behavioral baselines
- Authentication event visibility
- Local infrastructure telemetry

Any future endpoint expansion would continue following the platform's bounded ingest and local-first operational design principles.

Advanced Detection Workflows

The current detection architecture combines signature analysis, telemetry freshness validation, anomaly detection, and threat hypothesis generation.

Future detection enhancements may include:

- Behavioral baselining
- Traffic pattern anomaly scoring
- Historical deviation analysis
- Multi-source correlation rules
- Dynamic confidence evaluation
- Risk prioritization workflows
- Detection tuning automation
- Investigation-assisted detection refinement

The platform roadmap intentionally preserves human review and evidence visibility even as detection sophistication increases.

Long-Term Evidence Retention

Current persistence workflows focus on balancing operational usability with local infrastructure practicality. Future enhancements may expand long-term evidence retention strategies.

Potential enhancements may include:

- Cold storage workflows
- Historical archive pipelines
- Long-term investigation indexing
- Retention policy automation
- Evidence export tooling
- Historical telemetry summarization
- Tiered retention architecture

These enhancements would improve historical visibility while preserving the platform's structured evidence model.

Workflow Automation & Orchestration

The environment currently emphasizes operator-controlled workflows and bounded AI assistance. Future enhancements may introduce carefully constrained automation capabilities.

Potential automation workflows may include:

- Automated enrichment refresh
- Workflow orchestration
- Investigation assistance
- Notification pipelines
- Dashboard lifecycle automation
- Operational reporting
- Evidence packaging
- Remediation recommendation workflows

Any future automation would remain subject to the project's operational governance principles emphasizing reviewability, transparency, and bounded execution.

Reporting & Documentation Enhancements

Future operational enhancements may also improve reporting and documentation workflows.

Potential additions may include:

- Executive operational summaries
- Infrastructure status reporting
- Threat activity summaries
- CPE documentation workflows
- Detection effectiveness reporting
- Telemetry coverage analysis
- Investigation reporting
- Operational metrics dashboards

These capabilities would improve operational communication and long-term workflow continuity.

Infrastructure Scalability & Portability

The current Docker Compose deployment model supports local operational deployment and modular service management. Future architectural enhancements may improve portability and scalability.

Potential future directions may include:

- Multi-host deployment models
- Container orchestration improvements
- Distributed telemetry pipelines
- Externalized storage workflows
- Runtime redundancy
- Service scaling strategies
- Portable deployment profiles
- Infrastructure-as-code refinement

The architecture intentionally preserves modular service boundaries to support future scalability without requiring complete platform redesign.

Dashboard & Visualization Expansion

Grafana currently serves as the primary operational visualization layer for the environment. Future enhancements may expand dashboard capabilities through:

- Additional operational views
- Investigation-focused dashboards
- Historical trend visualizations
- Detection tuning panels
- Workflow analytics
- Infrastructure topology mapping
- Investigation timelines
- Correlation-focused visualizations

The roadmap continues emphasizing operational usability and evidence visibility rather than dashboard complexity for its own sake.

Continued AI-Assisted Engineering

Future engineering workflows are expected to continue leveraging bounded AI-assisted implementation support while preserving:

- Human approval
- Deployment governance
- Runtime validation
- Operational accountability
- Infrastructure transparency

Potential future improvements may include:

- Enhanced operational summarization
- Faster troubleshooting assistance
- Documentation generation
- Configuration review workflows
- Detection analysis support
- Investigation guidance improvements

The platform roadmap intentionally treats AI assistance as an engineering acceleration mechanism rather than an autonomous operational authority.

Long-Term Vision

The long-term vision for ZillaSIEM™ is not to become a generalized enterprise security platform, but rather to evolve into a mature, maintainable, evidence-oriented operational telemetry environment capable of supporting:

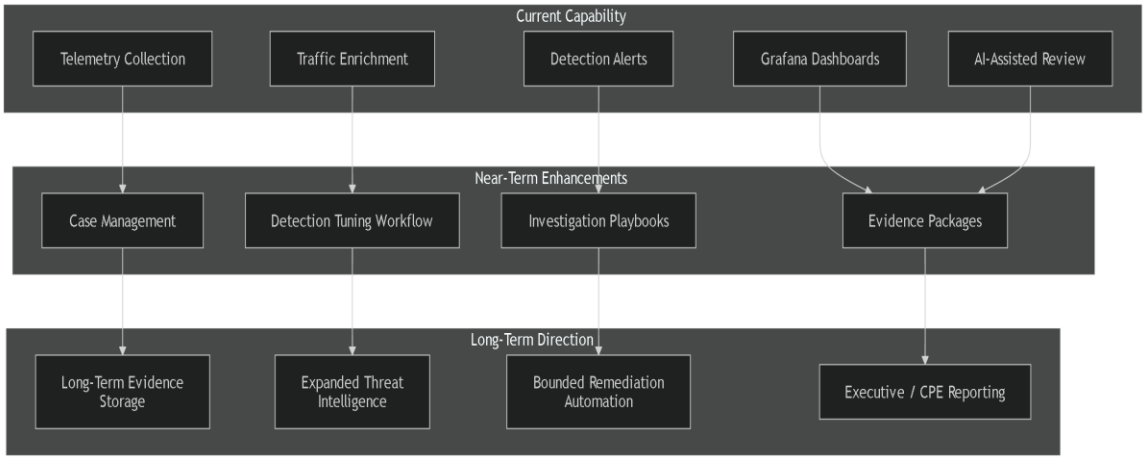
- Local network visibility
- Structured investigations
- Detection engineering
- Operational observability
- Contextual analysis
- AI-assisted workflows
- Human-guided review

- Continuous iterative improvement

within a modular and operator-controlled architecture.

The roadmap therefore prioritizes sustainable operational maturity, maintainable engineering practices, and evidence-driven workflows over feature expansion without architectural discipline.

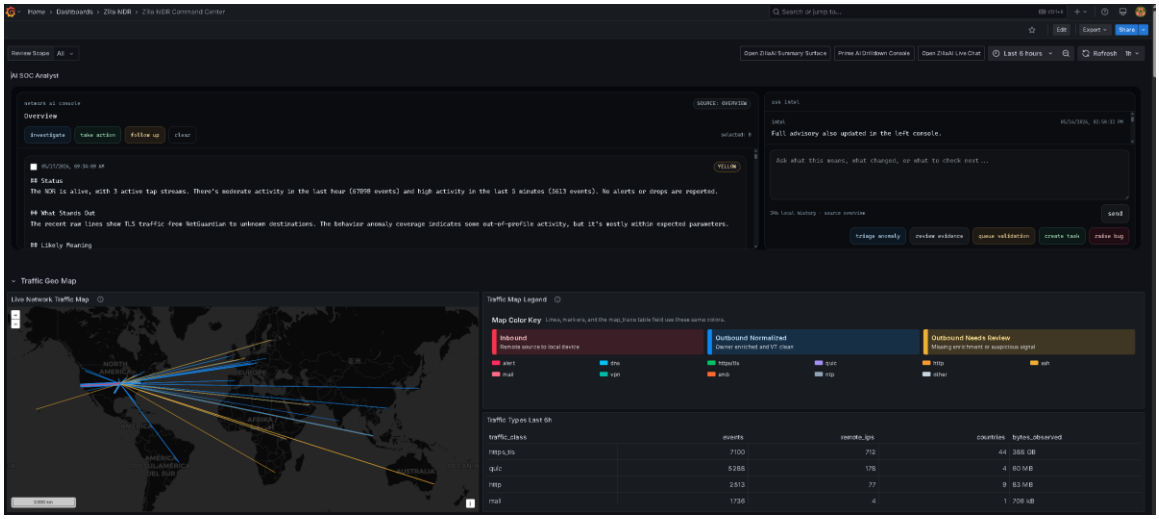
As the environment evolves, the foundational architectural principles established throughout the platform — modularity, bounded exposure, evidence preservation, operational visibility, and human-governed workflows — are intended to remain central to future development and operational refinement.

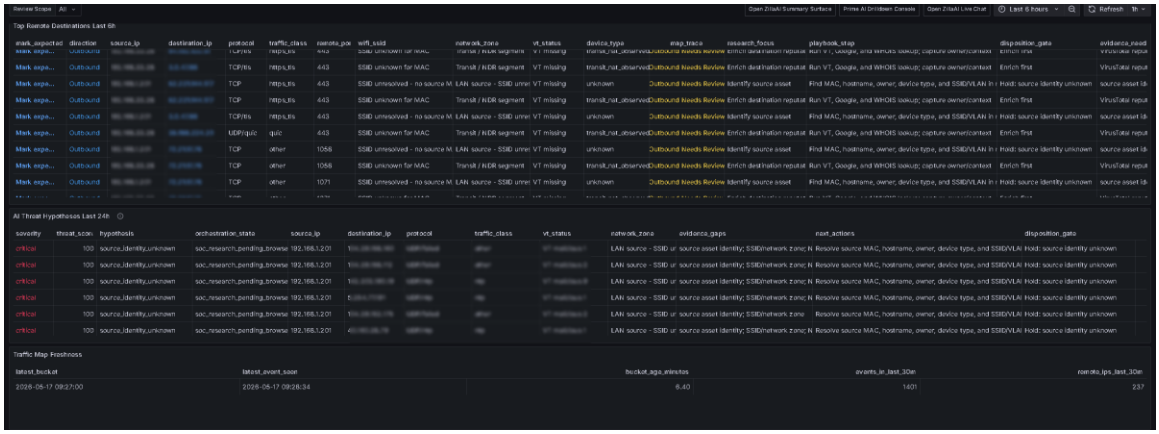


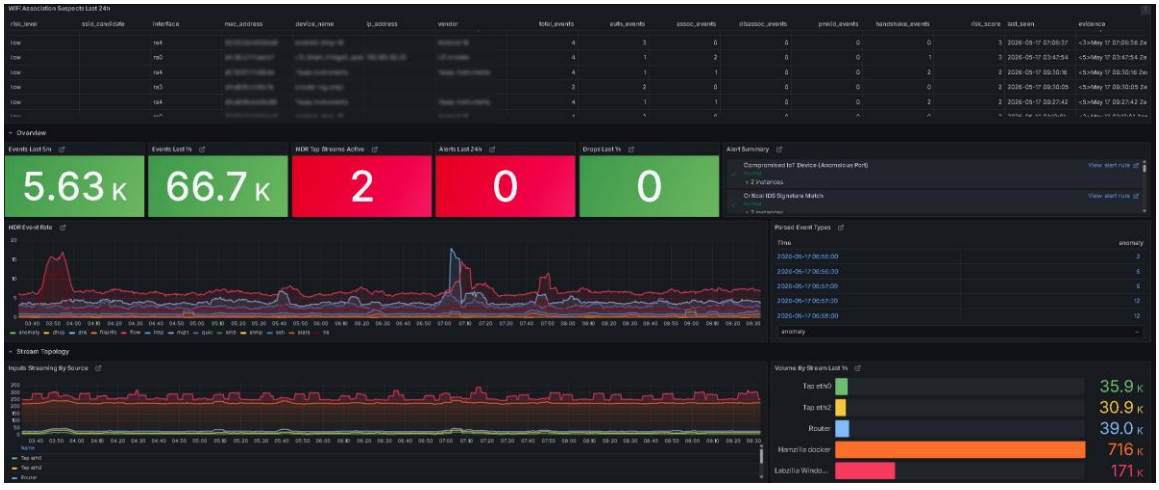
ZillaSIEM™ Roadmap & Capability Evolution

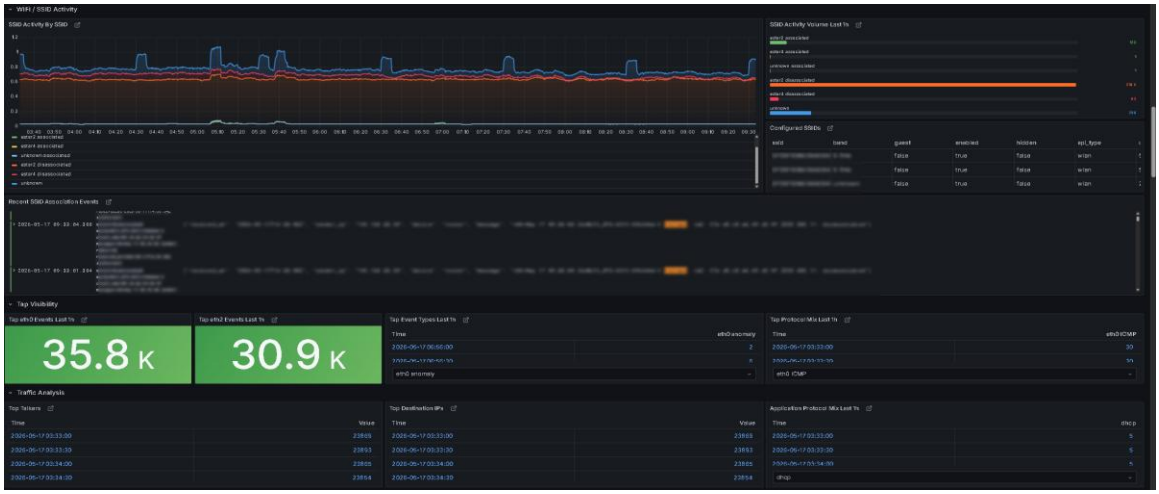
Appendix A — Dashboard Screenshots

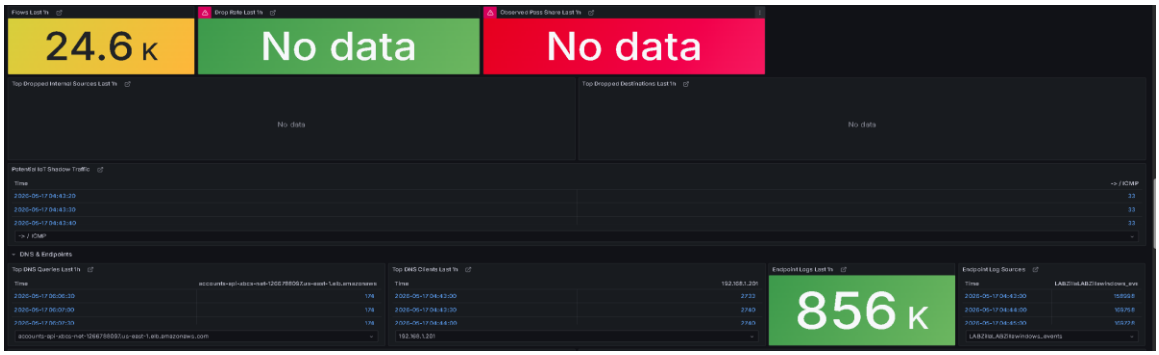
The following placeholders are reserved for sanitized operational dashboard captures and reconstructed publication-safe visuals.

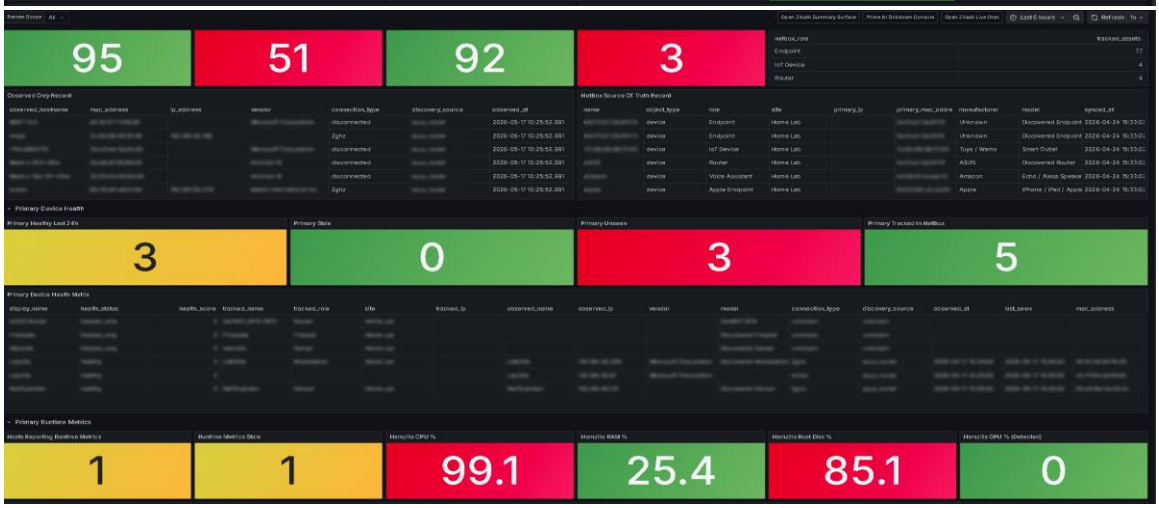


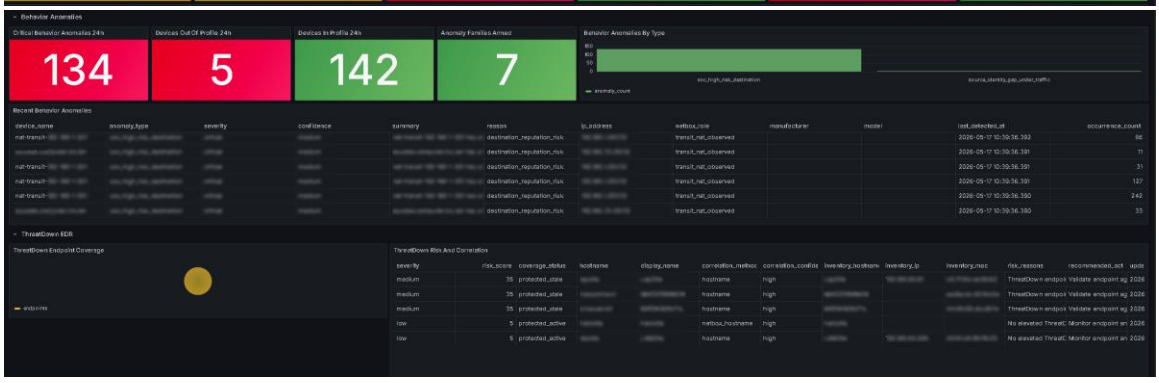












Appendix B — Diagram References

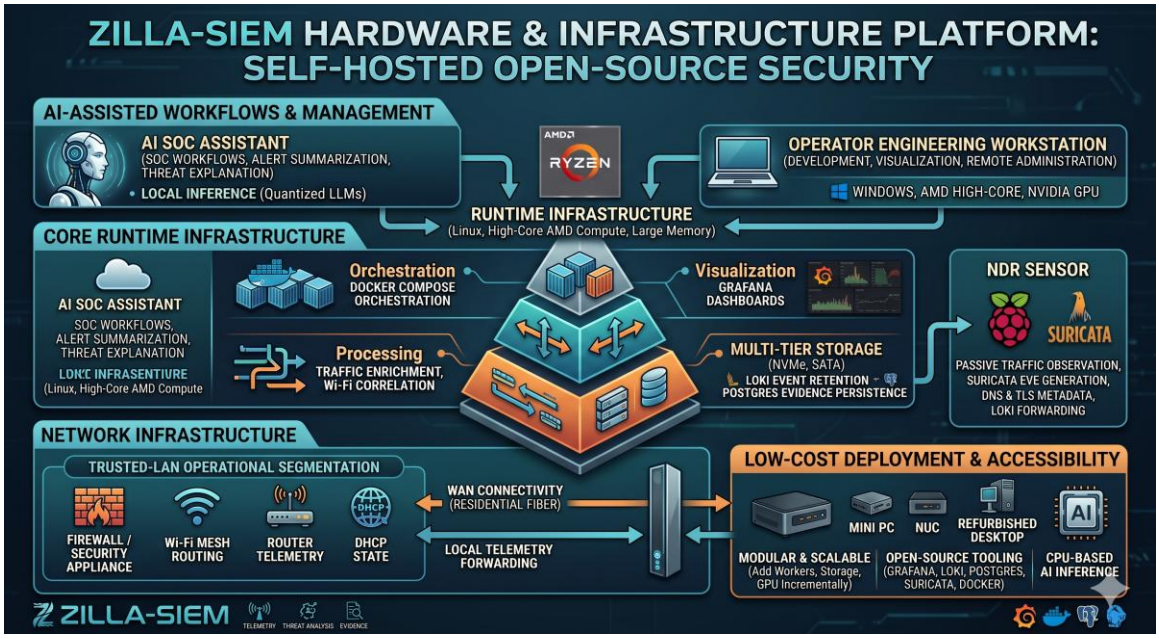
#	Section	Mermaid Diagram Name	Purpose / Description
1	Opening / Introduction	**ZillaSIEM™ Whitepaper Hero Graphic**	Executive-style overview showing telemetry, processing, detection, operations, AI-assisted review, and human validation.
2	3. Architecture Overview	**ZillaSIEM™ Platform Architecture**	High-level system architecture showing telemetry sources, ingestion, data layer, Grafana, ZillaAI, and analyst workflow.
3	4. Telemetry & Data Pipeline	**ZillaSIEM™ Telemetry & Evidence Pipeline**	Shows how raw telemetry becomes normalized, enriched, correlated, persisted, and surfaced for review.
4	5. Runtime Service Architecture	**ZillaSIEM™ Runtime Service Architecture**	Shows Docker Compose services, workers, dependencies, persistent volumes, and local runtime components.
5	6. Trust Boundaries & Exposure Model	**ZillaSIEM™ Trust Boundaries & Exposure Model**	Shows trusted LAN, Hamzilla local runtime boundary, operator-controlled engineering boundary, and no direct public SIEM exposure.
6	7. Threat Enrichment & Correlation	**ZillaSIEM™ Threat Enrichment & Correlation Pipeline**	Shows source attribution, NAT/Wi-Fi/inventory correlation, destination enrichment, and structured evidence outputs.
7	8. Detection & Alerting	**ZillaSIEM™ Detection & Alerting Pipeline**	Shows telemetry inputs, detection logic, severity/confidence evaluation, alerting, and analyst review.

• 8	• 9. Analyst Workflow	• **ZillaSIEM™ Analyst Workflow Pipeline**	• Shows detection signals moving through evidence preparation, hypothesis review, disposition, AI-assisted review, and human decision.
• 9	• 10. Networking & Exposure	• **ZillaSIEM™ Networking & Exposure Model**	• Shows ports, loopback/local services, trusted-LAN access, syslog ingest, endpoint ingest, and avoided public exposure.
• 10	• 11. Persistence & Retention	• **ZillaSIEM™ Persistence & Retention Model**	• Shows Loki raw event storage, Postgres evidence tables, Grafana state, provisioning, and retention-oriented data separation.
• 11	• 12. Operational Validation	• **ZillaSIEM™ Operational Validation & Health Monitoring**	• Shows health checks, telemetry freshness, Docker state, query validation, reboot checks, alerts, and operator review.
• 12	• 13. AI-Assisted Engineering Workflow	• **AI-Assisted Engineering & Validation Workflow**	• Shows human goals, Codex-assisted inspection/generation, controlled execution, BridgerW, runtime validation, and approval boundaries.
• 13	• 14. Incident Investigation Workflow	• **ZillaSIEM™ Incident Investigation Workflow**	• Shows investigation triggers, evidence review, scope/confidence/gaps, disposition, and operational outcomes.
• 14	• 15. Repository Ownership Model	• **ZillaAI / ZillaSIEM™ Repository Ownership Model**	• Shows separation between ZillaAI operator workflow ownership and ZillaSIEM™ telemetry/runtime ownership.
• 15	• 16. Roadmap & Future Enhancements	• **ZillaSIEM™ Roadmap & Capability Evolution**	• Shows current capabilities, near-term enhancements, and

			longer-term future-state direction.

Appendix C — Hardware & Infrastructure Platform

The ZillaSIEM™ environment was developed and operated using a locally managed hardware and network stack designed to support telemetry collection, network detection and response (NDR), operational monitoring, structured evidence generation, and AI-assisted analyst workflows within a trusted private-network deployment model.



The infrastructure architecture intentionally emphasized:

- Local operational ownership
- Trusted-LAN deployment
- Modular infrastructure growth
- Continuous telemetry visibility
- Structured evidence retention
- AI-assisted operational workflows
- Low-friction engineering iteration

Rather than relying on cloud-native infrastructure or externally hosted SIEM services, the environment was engineered as a self-hosted operational telemetry platform using

commodity and prosumer hardware integrated through Docker-based orchestration and locally managed network infrastructure.

C.1 Runtime Infrastructure

The primary operational runtime environment hosts the ZillaSIEM™ and ZillaAI stack and serves as the central telemetry-processing and orchestration platform for the environment.

Operational responsibilities include:

- Docker Compose orchestration
- Grafana dashboards
- Loki event retention
- Postgres evidence persistence
- Traffic enrichment workflows
- Wi-Fi correlation analysis
- Threat hypothesis generation
- Runtime validation services
- AI-assisted operational workflows

The runtime environment is based on:

- Linux server infrastructure
- High-core-count AMD compute architecture
- Large-memory operational configuration
- NVIDIA GPU acceleration
- Multi-tier NVMe and SATA storage
- Wired high-speed network connectivity

The runtime platform was designed to support simultaneous telemetry processing, dashboard rendering, enrichment workflows, AI-assisted operations, and persistent evidence retention within a local-first deployment model.

C.2 Operator Engineering Workstation

The environment includes a dedicated engineering and operational workstation used for:

- VS Code development workflows

- AI-assisted engineering operations
- Runtime validation
- Dashboard review
- Remote administration
- Deployment management
- Troubleshooting activities
- Documentation workflows

The workstation platform includes:

- Windows-based engineering infrastructure
- High-core-count AMD compute hardware
- Large-memory configuration
- NVIDIA GPU acceleration
- High-capacity NVMe and archival storage

This system functions as the primary operational management and engineering interface for the broader environment.

C.3 NDR Sensor Platform

The environment incorporates a lightweight dedicated NDR telemetry sensor responsible for passive network-defense visibility and Suricata event generation.

Operational responsibilities include:

- Passive traffic observation
- Suricata EVE telemetry generation
- DNS visibility
- TLS metadata observation
- Network flow telemetry
- Loki telemetry forwarding

The NDR platform is based on:

- Raspberry Pi infrastructure

- Dedicated passive telemetry workflows
- Lightweight continuous operation
- Independent telemetry generation
- Dedicated network monitoring interfaces

The NDR sensor provides operational network visibility while remaining logically separated from the primary SIEM runtime environment.

C.4 Network Infrastructure

The ZillaSIEM™ environment operates within a segmented trusted local network architecture consisting of:

- Wi-Fi mesh routing infrastructure
- Dedicated firewall/security appliance infrastructure
- Trusted-LAN operational segmentation
- Local telemetry forwarding paths
- Structured syslog ingestion workflows

The network infrastructure provides:

- Router telemetry
- Syslog forwarding
- Device inventory visibility
- DHCP and association-state visibility
- Network segmentation
- Firewall telemetry
- Trusted-LAN operational access

The architecture intentionally minimizes unnecessary public exposure while preserving operational visibility and telemetry continuity within the private environment.

C.5 WAN & Internet Connectivity

The environment uses residential fiber internet connectivity as the upstream WAN provider supporting:

- Internet access

- External enrichment lookups
- Operational package retrieval
- Threat intelligence workflows
- Remote administrative connectivity

The WAN environment remains separated from internal telemetry infrastructure through dedicated firewall and routing controls.

C.6 Docker Runtime & Service Architecture

The operational environment uses Docker Compose for runtime orchestration and service management.

Containerized services include:

- Grafana
- Loki
- Postgres
- ZillaSIEM™ APIs
- Syslog forwarders
- Traffic analysis workers
- Wi-Fi correlation services
- Threat analysis services
- Runtime monitoring services

The containerized deployment model supports:

- Service isolation
- Repeatable deployments
- Modular infrastructure management
- Runtime portability
- Operational reproducibility
- Independent service lifecycle control

This architecture simplified iterative engineering workflows and improved operational maintainability across the environment.

C.7 Operational Network Topology

The infrastructure operates within a trusted private-network architecture emphasizing:

- Trusted-LAN operational access
- Minimal public exposure
- Local telemetry ownership
- Internal telemetry processing
- Bounded operational APIs
- Segmented infrastructure workflows

The operational topology separates:

- WAN connectivity
- Firewall infrastructure
- Router and Wi-Fi infrastructure
- SIEM runtime services
- NDR telemetry generation
- Analyst and engineering systems

This layered architecture supports local operational visibility while minimizing unnecessary exposure of telemetry infrastructure and evidence systems.

C.8 Infrastructure Design Philosophy

The hardware and infrastructure architecture reflects several broader engineering principles that shaped the project:

- Local operational ownership
- Practical observability engineering
- Modular service composition
- Evidence-oriented telemetry retention
- Trusted-LAN operational workflows
- Human-governed infrastructure management
- AI-assisted engineering acceleration

The resulting environment demonstrates that meaningful SIEM, NDR, telemetry enrichment, and analyst workflow capabilities can be developed using locally managed infrastructure and modular open-source tooling without requiring large-scale enterprise infrastructure or externally hosted operational dependencies.

C.9 Whitepaper Summary

The ZillaSIEM™ platform was developed and validated on a real operational home-lab infrastructure environment rather than a purely simulated deployment model. The environment combines dedicated runtime infrastructure, engineering workstations, passive NDR telemetry sensors, firewall and router telemetry sources, and Docker-orchestrated SIEM services into a unified operational telemetry platform.

Together, these systems support:

- Telemetry collection
- Network visibility
- Threat enrichment
- Structured evidence persistence
- Operational dashboards
- AI-assisted workflows
- Detection engineering
- Analyst-oriented review processes

within a trusted local operational environment designed for iterative engineering, operational validation, and evidence-driven security telemetry workflows.

Lets also discuss the low cost investment on setting somethink like this up, because we likely could have run Grafana on a substantially lower model PC and used smaller quantized LLM Model to serve as SOC agent.

C.10 Cost Efficiency & Accessible Deployment Model

One of the more important architectural observations from the ZillaSIEM™ project is that meaningful SIEM, NDR, telemetry enrichment, and analyst workflow capabilities can be implemented without requiring enterprise-scale infrastructure investment.

Although the development environment utilized high-performance workstation hardware to support simultaneous engineering, AI-assisted workflows, GPU experimentation, telemetry processing, and operational development activities, the core ZillaSIEM™ architecture itself was intentionally modular and lightweight enough to operate on substantially lower-cost infrastructure.

This distinction is important because the platform was engineered for:

- Architectural flexibility
- Modular deployment
- Local operational ownership
- Service separation
- Incremental scalability

rather than dependency on enterprise-grade compute environments.

C.10.1 Practical Minimum Infrastructure Requirements

Several core components of the environment have relatively modest operational requirements when deployed for small-scale or home-lab monitoring use cases.

Examples include:

Component	Practical Resource Profile
Grafana	Lightweight dashboarding workload
Loki	Moderate memory and storage dependent
Postgres	Low-to-moderate structured evidence workload
Syslog ingestion	Extremely lightweight
Suricata NDR	Moderate depending on traffic volume
Enrichment workers	CPU-oriented background processing
API services	Lightweight web-service workloads

For small environments, many of these services can operate effectively on:

- Small-form-factor PCs
- Mini PCs
- Refurbished enterprise desktops
- Intel NUC-style systems
- Low-power AMD systems
- Repurposed workstation hardware

The architecture intentionally avoids mandatory dependency on large-scale distributed infrastructure.

C.10.2 Separation Between Development Hardware & Runtime Requirements

The development environment utilized high-end hardware primarily because the system simultaneously supported:

- AI-assisted engineering workflows
- Local LLM experimentation
- GPU acceleration testing
- Documentation generation
- Runtime orchestration
- Dashboard rendering
- Development tooling
- Multi-service operational testing

However, the core SIEM runtime itself does not inherently require workstation-class hardware for smaller operational deployments.

For example:

- Grafana dashboards can operate on relatively low-resource systems
- Loki performs efficiently in modest local environments
- Postgres structured evidence storage scales effectively for small deployments
- Syslog ingestion has minimal resource overhead
- Lightweight enrichment workers can run on commodity CPUs

The project demonstrated that the operational telemetry architecture remains viable even when scaled down substantially.

C.10.3 Low-Cost AI-Assisted SOC Model

Another important architectural observation is that AI-assisted SOC workflows do not necessarily require large frontier-scale models or expensive GPU infrastructure.

Many operational analyst-assistance workflows can be supported using:

- Quantized local LLMs
- Smaller parameter models

- CPU-assisted inference
- Low-VRAM GPU acceleration
- Hybrid cloud/local inference strategies

Potential AI-assisted operational tasks suitable for lightweight models include:

- Alert summarization
- Threat explanation
- Dashboard interpretation
- Analyst note generation
- Workflow assistance
- Operational summarization
- Investigation guidance
- Basic enrichment interpretation

This creates opportunities for low-cost AI-assisted SOC experimentation without requiring enterprise AI infrastructure investments.

C.10.4 Example Low-Cost Deployment Scenarios

A practical low-cost deployment model could include:

Function	Example Low-Cost Approach
SIEM Runtime	Refurbished mini PC or small workstation
NDR Sensor	Raspberry Pi with passive monitoring
Firewall	Prosumer security appliance
Router Telemetry	Existing mesh/Wi-Fi infrastructure
Dashboards	Grafana on lightweight Linux host
AI SOC Assistant	Quantized local LLM
Storage	Commodity SATA/NVMe drives
Orchestration	Docker Compose

This significantly lowers the barrier to entry for:

- Home-lab operators
- Students
- Security researchers
- Small organizations
- Independent analysts
- Blue-team experimentation
- Detection engineering practice

C.10.5 Modular Scalability Philosophy

The architecture was intentionally designed so infrastructure capabilities could scale incrementally.

Examples include:

- Adding additional telemetry workers
- Expanding storage capacity
- Introducing GPU acceleration
- Scaling enrichment pipelines
- Separating workloads across hosts
- Adding additional sensors
- Expanding dashboard environments

This modular approach allows operators to begin with modest infrastructure investments and expand operational capability over time.

C.10.6 Accessibility Of Modern Open-Source Security Tooling

The project also demonstrates how modern open-source tooling has significantly lowered the cost of building operational visibility environments.

Capabilities once associated primarily with enterprise SOC environments are now accessible using:

- Open-source telemetry tooling
- Containerized deployment models

- Commodity hardware
- Local AI inference
- Community-supported observability platforms

Examples include:

- Grafana
- Loki
- Postgres
- Suricata
- Docker Compose
- Local LLM inference frameworks

This ecosystem enables meaningful experimentation and operational learning without requiring enterprise procurement cycles or large infrastructure budgets.

C.10.7 Operational Philosophy

A major takeaway from the ZillaSIEM™ project is that effective operational visibility is no longer exclusively dependent on expensive enterprise infrastructure.

Instead, modern observability and telemetry workflows can increasingly be built using:

- Modular open-source tooling
- Locally managed infrastructure
- Commodity compute hardware
- Lightweight AI-assisted workflows
- Incremental operational scaling

This makes advanced SIEM, NDR, and analyst-assistance experimentation substantially more accessible to independent operators, researchers, students, and small-scale environments while still preserving meaningful architectural and operational sophistication.

Appendix D — NIST SP 800-53 Rev. 5 Control Mapping Summary

This appendix provides a high-level mapping of the ZillaSIEM™ architecture, operational workflows, telemetry pipelines, detection capabilities, and evidence-oriented monitoring functions to applicable NIST SP 800-53 Rev. 5 control families and individual controls. The

mapping is intended to identify areas where the documented platform design appears to support, contribute to, or partially address control objectives relevant to SIEM, NDR, telemetry monitoring, operational visibility, and analyst workflow management.

The mapping is based exclusively on statements, workflows, architectural descriptions, operational models, and engineering practices explicitly documented within the reviewed white paper. No assumptions were made regarding undocumented implementations, unstated operational procedures, inherited controls, compensating safeguards, or implied governance processes.

This appendix does not constitute:

- A formal compliance determination
- A certification of control implementation
- An audit opinion
- Evidence of operational effectiveness
- Validation of control maturity
- A complete authorization assessment

Instead, the mapping identifies where the documented architecture appears to align with the intent of specific NIST SP 800-53 Rev. 5 controls based on the described functionality and operational design.

The controls identified in this appendix primarily relate to:

- Audit logging and telemetry collection
- Continuous monitoring
- Detection and alerting
- Boundary protection
- Operational visibility
- Structured evidence retention
- Incident investigation workflows
- Configuration and deployment governance
- Runtime validation
- Service isolation

- AI-assisted engineering governance

Several mapped controls are only partially addressed due to limited evidence regarding:

- Formal policy governance
- Administrative procedures
- Role-based access governance
- Operational testing
- Retention enforcement
- Detection lifecycle management
- Encryption assurance
- Incident escalation procedures
- Audit evidence generation
- Control effectiveness validation

Where the white paper demonstrates architectural intent but lacks sufficient implementation evidence, the mapping should be interpreted as "control-aligned" rather than "fully implemented."

This appendix is intended to support:

- Security architecture review
- Preliminary governance analysis
- Procurement evaluation
- Risk assessment preparation
- Gap analysis activities
- POA&M development
- Future compliance alignment efforts

The following table summarizes the NIST SP 800-53 Rev. 5 controls that appear to be addressed, supported, or partially implemented by the documented ZillaSIEM™ architecture and associated operational workflows.

Control	Control Name	White Paper Basis
AC-3	Access Enforcement	Trusted-LAN access, bounded service exposure, controlled API access
AC-4	Information Flow Enforcement	Separation of telemetry processing, APIs, dashboards, and operator workflows
AC-6	Least Privilege	Loopback-oriented Loki/Postgres and minimized internal service exposure
AU-2	Event Logging	Router, firewall, Suricata, host, runtime, and infrastructure telemetry collection
AU-3	Content of Audit Records	Structured telemetry, event enrichment, attribution, and evidence records
AU-6	Audit Record Review, Analysis, and Reporting	Grafana dashboards, analyst workflows, alert review, and operational drilldowns
AU-8	Time Stamps	Implied by event telemetry and evidence timelines; implementation detail not fully specified
AU-9	Protection of Audit Information	Local-first telemetry ownership and restricted exposure of raw telemetry stores
AU-11	Audit Record Retention	Loki raw event retention and Postgres structured evidence persistence
AU-12	Audit Record Generation	Syslog, Suricata, runtime metrics, worker logs, and infrastructure telemetry
CA-7	Continuous Monitoring	Telemetry freshness, service health checks, dashboard validation, and alerting
CM-2	Baseline Configuration	Docker Compose deployment model and repo-managed configuration/provisioning
CM-3	Configuration Change Control	Human approval for high-impact infrastructure, exposure, and deployment changes
CM-6	Configuration Settings	Loopback-oriented services, bounded ports, host firewall controls
CM-7	Least Functionality	Avoidance of broad public exposure and minimized service exposure

CP-2	Contingency Plan	Partially addressed through post-reboot validation and recovery-oriented checks
CP-9	System Backup	Persistence model discussed, but backup implementation is Information not available
IA-2	Identification and Authentication	Token-authenticated runtime metric ingest; broader IAM details are Information not available
IR-4	Incident Handling	Incident investigation workflow, evidence review, hypothesis validation, operational outcomes
IR-5	Incident Monitoring	Detection, alerting, telemetry monitoring, and investigation workflows
IR-6	Incident Reporting	Partially addressed through roadmap reporting and documentation enhancements
RA-5	Vulnerability Monitoring and Scanning	Threat enrichment, detection engineering, IDS signals, suspicious destination review
RA-7	Risk Response	Analyst disposition, remediation workflows, and operational outcome handling
SA-3	System Development Life Cycle	AI-assisted engineering with human review, deployment governance, and validation
SA-8	Security and Privacy Engineering Principles	Modular design, separation of concerns, bounded exposure, evidence preservation
SA-10	Developer Configuration Management	Repository ownership model and repo-managed provisioning/configuration
SC-7	Boundary Protection	Trusted-LAN model, minimized public exposure, firewall controls, internal service isolation
SC-8	Transmission Confidentiality and Integrity	Partially addressed through bounded trusted-LAN ingest; encryption detail is Information not available
SC-39	Process Isolation	Containerized service separation and modular Docker Compose runtime
SI-4	System Monitoring	Core control addressed: SIEM/NDR telemetry, detection, alerting, dashboards, health monitoring

SI-6	Security and Privacy Function Verification	Operational validation, query validation, worker monitoring, dashboard validation
SI-7	Software, Firmware, and Information Integrity	Partially addressed through repository governance and controlled deployment validation
SR-3	Supply Chain Controls and Processes	Partially addressed through open-source/custom component awareness and bounded AI-assisted engineering
SR-11	Component Authenticity	Partially addressed; detailed authenticity validation is Information not available

Appendix E — SIEM Control Gap Analysis

This appendix documents identified control gaps and operational assurance deficiencies associated with the current implementation and documented architecture of the ZillaSIEM™ monitoring environment, specifically limited to firewall operations, Network Detection and Response (NDR), TAP/SPAN visibility architecture, and SIEM capabilities.

The purpose of this appendix is to identify areas where operational capability is present but formal governance, validation, procedural maturity, audit evidence, or measurable assurance mechanisms are not fully demonstrated within the reviewed white paper and supporting architectural documentation. The analysis is based solely on documented architectural descriptions, operational workflows, telemetry pipelines, and stated engineering practices. It does not assume undocumented controls, implied operational procedures, or unstated governance mechanisms.

The identified gaps should not be interpreted as evidence of control failure or absence of operational capability. In many cases, the platform demonstrates strong architectural alignment with modern monitoring, telemetry enrichment, operational visibility, and analyst workflow principles. However, several areas lack sufficient documentation or evidence required to support formal audit validation, repeatable governance assurance, or regulatory defensibility.

The gaps identified in this appendix primarily relate to:

- Formal governance and operationalization
- Detection lifecycle management
- Visibility coverage validation
- Administrative control assurance

- Retention and evidence governance
- Configuration management maturity
- Incident escalation integration
- Monitoring integrity validation
- Segmentation and exposure assurance

This appendix is intended to support:

- Security architecture review
- Governance maturity assessment
- Operational hardening activities
- Audit preparation
- Control formalization efforts
- Future POA&M development
- Risk treatment planning

The observations contained herein are aligned to applicable NIST SP 800-53 Rev. 5 control families and reflect a deterministic review of the documented implementation scope for firewall, NDR, TAP/SPAN, and SIEM operational capabilities.

Gap Area	**Relevant NIST Controls**	**Gap Description**	**Severity**
Formal Firewall Rule Governance	CM-3, CM-6, SC-7	Firewall rule review, approval, recertification, and change governance processes are not formally described	High
Network Segmentation Enforcement Evidence	SC-7, AC-4	Trusted-LAN segmentation is discussed, but segmentation architecture and enforcement validation are not evidenced	High
East-West Visibility Coverage	SI-4, SC-7	White paper emphasizes perimeter and telemetry ingest visibility, but east-	Moderate

		west internal network monitoring coverage is not fully demonstrated	
TAP/SPAN Validation Procedures	CA-7, SI-4	TAP/SPAN integrity monitoring, packet-loss validation, and coverage assurance are not documented	High
NDR Sensor Placement Governance	SI-4, SC-7	Sensor placement methodology and monitored boundary coverage are not formally defined	Moderate
Encrypted Traffic Inspection Strategy	SI-4, SC-8	TLS visibility limitations and encrypted traffic inspection strategy are not operationally defined	Moderate
Firewall Logging Completeness	AU-2, AU-12	Firewall telemetry ingestion is described, but completeness validation and logging standards are not evidenced	Moderate
Log Integrity Protection	AU-9	Tamper resistance, integrity validation, and chain-of-custody controls for SIEM evidence are not documented	High
SIEM Retention Governance	AU-11	Retention duration, archival policy, and evidence lifecycle governance are not formally defined	Moderate
Detection Rule Lifecycle Management	SI-4, CM-3	Detection engineering is discussed, but formal rule testing, promotion, rollback, and approval workflows are not evidenced	High
False Positive / Tuning Governance	SI-4	No formal detection tuning governance or analyst feedback workflow is defined	Moderate
Correlation Rule Validation	SI-4, SI-6	Threat correlation logic exists, but validation and QA procedures are not documented	Moderate
Time Synchronization Assurance	AU-8	Event timestamp synchronization and authoritative time-source governance are not described	Moderate
Incident Escalation Integration	IR-4, IR-5	Detection and analyst review exist, but escalation procedures and operational	Moderate

		response integration are not formally defined	
Firewall HA / Resilience Validation	CP-10, SC-5	Firewall failover testing, resiliency validation, and degraded-mode operations are not evidenced	Moderate
SIEM Capacity & Scaling Governance	SI-4, AU-4	Log ingestion saturation thresholds, storage exhaustion handling, and scaling strategy are not formally documented	Moderate
Packet Capture Retention Strategy	AU-11, SI-4	TAP/NDR packet retention scope and storage governance are not defined	Moderate
Telemetry Source Authentication	IA-3, AU-12	Syslog trust assumptions are discussed, but authenticated telemetry source validation is not demonstrated	High
Administrative Access Governance	AC-2, AC-6	Firewall/SIEM/NDR administrative access review and privileged role governance are not evidenced	High
Detection Coverage Metrics	CA-7, SI-4	No measurable detection coverage KPIs or visibility coverage metrics are documented	Moderate
SIEM Evidence Classification	MP-4, AU-9	Classification and handling requirements for retained telemetry/evidence are not formally defined	Moderate
NDR Signature Governance	SI-3, SI-4	Signature update validation, testing, and rollback governance are not operationally documented	Moderate
Operational Runbooks	IR-4, CP-2	Analyst workflows exist conceptually, but formal runbooks and repeatable response procedures are not evidenced	Moderate
Firewall Configuration Backup Governance	CP-9, CM-2	Backup validation and restoration testing for firewall/network configurations are not described	Moderate
SOC Separation of Duties	AC-5	Operational role separation between administrators, analysts, and	Moderate

		engineering workflows is not formally defined	

