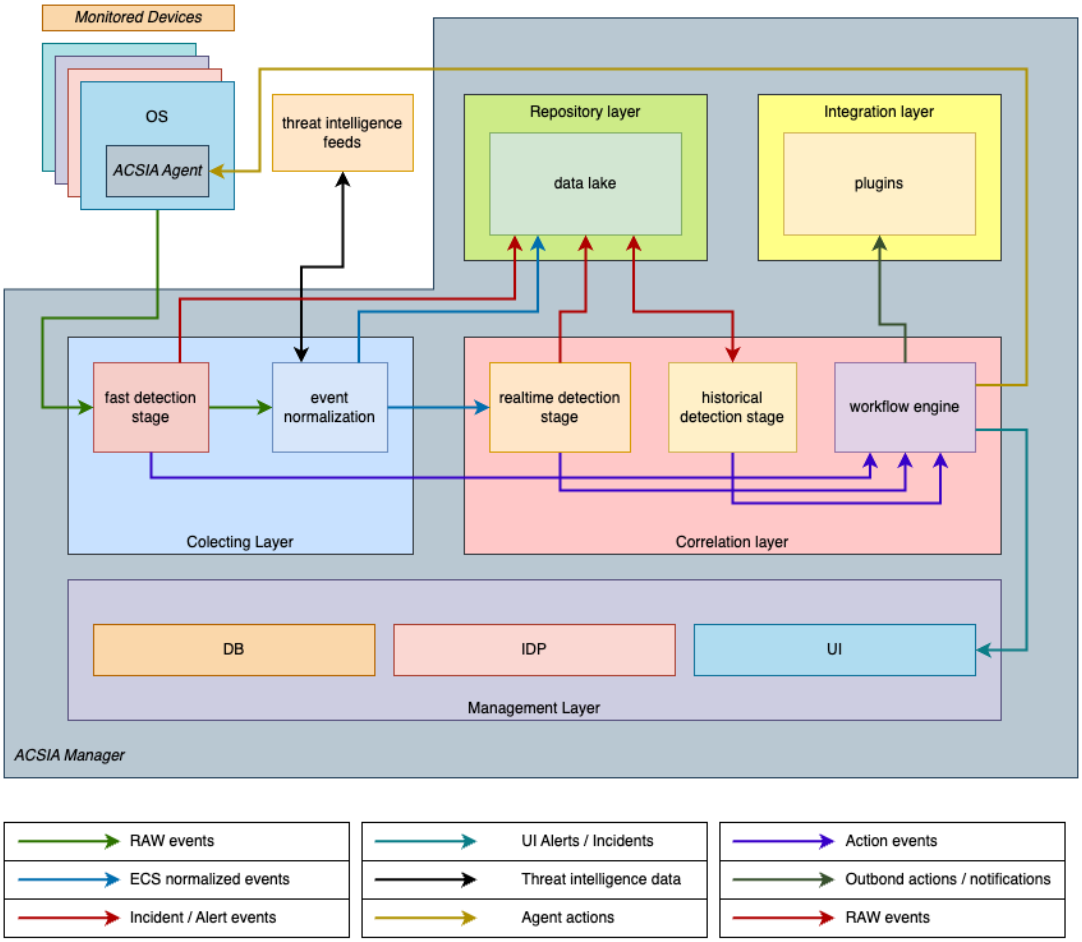
ACSIA: Multi-Layered Security Architecture for Comprehensive Threat Detection and Response

ACSIA SOS (Security Operation Solutions) is a robust cybersecurity solution that delivers exceptional protection against modern cyber threats. Its core lies in a meticulously designed, multi-layered detection system powered by advanced threat intelligence and cutting-edge technologies.

Key Architectural Components:

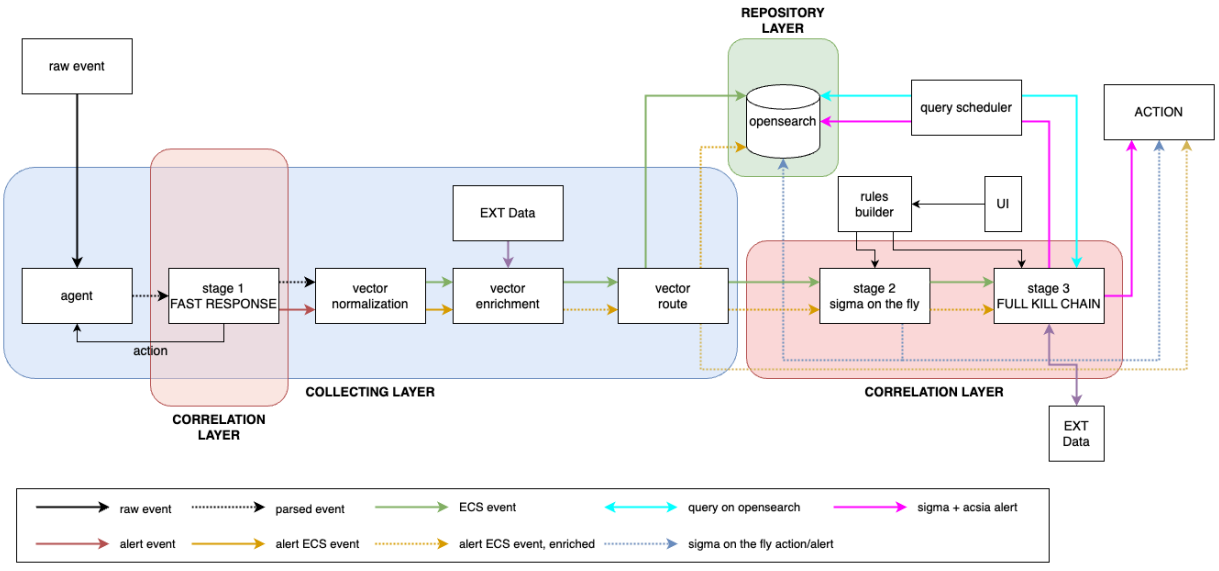
- **Five-Layered Architecture:** ACSIA leverages a five-layered architecture to provide comprehensive security functionalities, catering to diverse needs while maintaining the highest performance standards.



- **Collection Layer:** This layer forms the backbone, gathering and normalizing logs from various sources using the Elastic Common Schema (ECS) for consistency. Wazuh and Vector play a crucial role in this process, ensuring seamless aggregation and normalization across multiple sources. Additionally, this layer enriches data with Indicators of Compromise (IoC) and Indicators of Attack (IoA) for enhanced contextual understanding.

- **Correlation Layer:** This layer features a multi-stage detection engine with advanced capabilities, including Sigma rule compatibility for custom detection mechanisms. It also houses a remediation engine empowered by customizable workflows, enabling tailored responses to identified threats.

  - **Detection Engine:** This sophisticated engine continuously analyzes network traffic, system logs, and anomalous behaviors to detect and alert on all types of security threats. It employs advanced algorithms and pattern recognition to identify suspicious activities like intrusions, malware, and enumeration attempts. The engine operates in three distinct stages:



- **Stage 01:** The stage 01 leverages Wazuh signatures for immediate action, swiftly identifying known threats and triggering response measures. It is the fastest detection mechanism of the ACSIA suite and it operates before the collecting layer, as soon as logs and events hit the platform.
- **Stage 02:** The stage 02 boasts compatibility with Sigma rules, a meta-language that allows for the definition of customized detection mechanisms or the utilization of readily accessible predefined rules from the web. This feature empowers users to create tailored detection strategies suited to their specific security needs or leverage community-developed rules for comprehensive threat detection coverage. By integrating Sigma rules, our engine enhances its versatility and effectiveness in identifying and mitigating cybersecurity threats across diverse environments. Using Sigma rules in real-time against a flow of events, this stage enables dynamic detection of complex attack patterns as they unfold.
- **Stage 03:** The stage 03 employs a time-span-based approach to reconstruct the full kill chain, allowing for comprehensive threat analysis and strategic response planning. By analyzing the information gathered from various sources, the correlation layer identifies potential threats and triggers tailored response actions. This enables customers to customize the response actions automating SOC incident response action.

- **Workflow Engine:** Working alongside the detection engine, this component facilitates swift and effective responses through user-defined workflows. This allows organizations to orchestrate remediation actions aligned with their specific security protocols and compliance requirements.

- **Repository Layer:** Built upon OpenSearch, this layer serves as a vast data lake for storing security data. OpenSearch offers a robust and scalable platform for efficient management and querying of large datasets, making it ideal for storing diverse logs and telemetry data.

- **Management Layer:** This serves as the control center, offering a comprehensive 360-degree view of activities across the customer's digital infrastructure. The user interface (UI) supports various authentication methods and access control in a multi-tenant environment.

- **Integration Layer:** This layer acts as the interface between ACSIA and the environment, allowing for extensibility through plugins for various purposes like FW orchestration, SIEM integration, etc.

- **Microservice Components and Technologies:** ACSIA leverages a microservices architecture, offering modularity, scalability, and continuous delivery. Here's a breakdown of key components and their technologies:

  - **Collecting Layer:**

- Xdrplus-vector-aggregator: Built on Vector, this microservice performs tasks like event acceptance from Wazuh manager, standardization using ECS, geolocation enrichment, event forwarding to the detection engine and alerts to the action service, and data transmission to OpenSearch for storage and indexing. Additionally, it tags known malicious IPs based on the Xdrplus malicious IPs database.
- Xdrplus-wazuh-manager: This microservice integrates Wazuh functionality, receiving events from co-located Wazuh agents, parsing and processing them, forwarding processed events to the vector aggregator, and identifying/raising alerts based on its internal XML meta-language.

- **Repository Layer:**

  - Xdrplus-opensearch: This is a standard OpenSearch image.
  - Xdrplus-opensearch Dashboards: This is a standard Opensearch dashboard.

- **Management Layer:**

  - Identity and Access Management:
    - Xdrplus-iam: This Python service with Django framework and PySaml2 libraries acts as a SAML2 IDentity Provider and identity manager, enabling user authentication and multi-tenancy management.
    - Xdrplus-iam-db: This PostgreSQL database serves as the data repository for xdrplus-iam.

- Xdrplus-iam-nginx: This standard Nginx image handles IAM incoming requests.

- Reverse Proxy: Xdrplus-nginx (standard Nginx image) manages incoming requests from users (via UI) and applications (agents and Wazuh clients).
- Messaging System: RabbitMQ acts as a message broker, employing a publisher-subscriber pattern to dispatch events from the collecting layer to the detection engine.
- User Interface:
  - Xdrplus-uife: This React-based microservice with Tailwind and Radix libraries exposes a UI for user, device, incident, general setting, network policy, detection rule, audit logging, and management functionalities.
  - Xdrplus-uibe: This NodeJS microservice with GraphQL serves as the backend for the UI, executing modifications and actions via REST calls to other microservices.
  - Xdrplus-uiredis: This Redis service handles user sessions.

- **Detection Engine:**

  - Xdrplus-stage-engine (Java): This microservice utilizes the Sigma Rule engine with Apache Beam for flexibility and scalability to detect anomalies and identify attacks/patterns, sending generated alerts to the workflow engine.

- Xdrplus-stage-inst-engine (Java): Similar to the above, it employs a time-span-based approach for comprehensive threat analysis and strategic response planning.
- Xdrplus-stage-instant (Python): This utilizes PySigma to run Sigma Rules in the OpenSearch environment in real-time.
- Xdrplus-stage-redis: This Redis service caches the latest events to identify attack types.
- Xdrplus-stage-services (Java): This microservice manages the detection engine and exposes APIs for Sigma rules, Opensearch Detector, and workflows.

- **Workflow Engine:**

  - Xdrplus-workflow-engine (NodeJS): This processes Xdrplus workflows and can call the actions service to execute actions.
  - Xdrplus-actions-service (NodeJS): This exposes a REST API for calling other APIs and executing actions on other components.

- **Data Management Layer:**

  - Xdrplus-das (Golang): This service exposes gRPC services for common application data like devices, network policies, tenant options, licenses, etc. It utilizes the Xdrplus-das-db microservice for data persistence.
  - Xdrplus-das-db (PostgreSQL): This standard database stores Xdrplus-das service data.

- Xdrplus-agentbe (Golang): This service acts as an interface with the device's agent, allowing agent registration, configuration retrieval, status updates, and agent upgrade triggering.

- **Agent Layer:**

  - Xdrplus-agent: This Golang application, packaged in OS installation bundles, has a plugin-based architecture where each plugin handles specific software configurations or actions like blocking malicious activities. It communicates with the Xdrplus-agentbe backend service for configuration and instructions. The communication is encrypted using ECDSA encryption with unique keys for each agent and enforced authentication.

This technical overview provides a comprehensive understanding of ACSIA's architecture and the key components that work together to deliver robust threat detection and response capabilities.