The Open Cybersecurity Alliance (OCA) Architecture

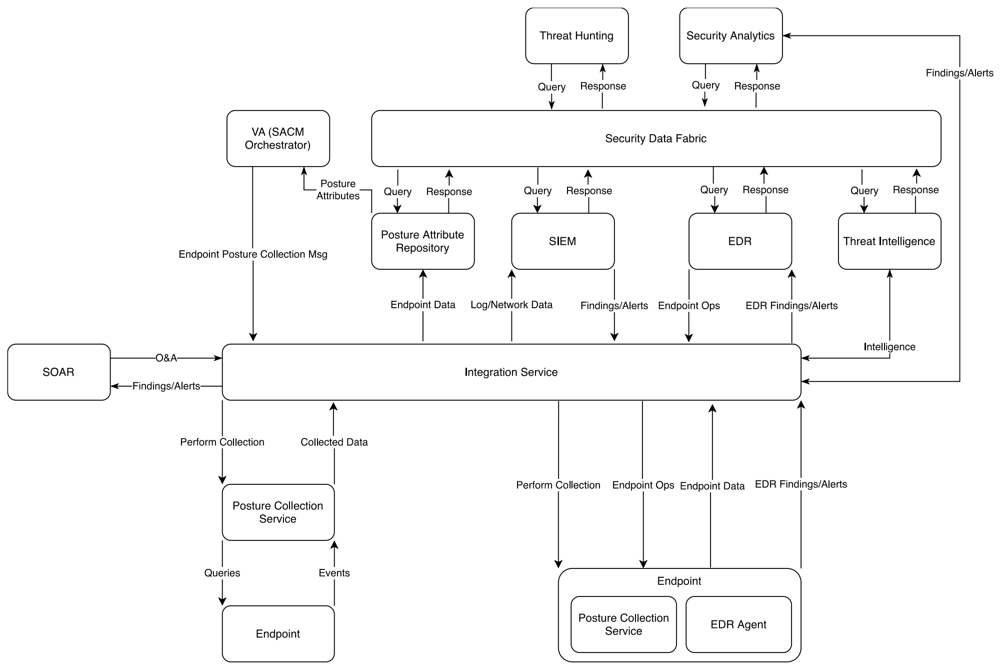
Objectives

The establishment of the OCA is built on the realization that security tools need to speak a common language so that interoperability can be achieved at the communication and data levels. The goal of the OCA is to enhance interoperability and collaboration around various different standards, tools, procedures and open-source libraries. Through this, organizations will be able to create a more sustainable approach to addressing the increasing volume and sophistication of security threats by being better able to identify, drill into and prioritize remediation more effectively

This document will provide an overview of the OCA architecture its key components and how collaboration among security products can be attained. Key elements of the architecture are:

Capability	Description	Technology
Common communications channel for collaboration	A Publish/Subscribe model to enable 1 to many communication	OpenDXL
Normalization of security information	Queries for Observable data	STIXShifter
Ontology for security communications	Common actions and response formats	OpenDXL Ontology
Support automated enterprise security assessment	Support continuous monitoring of enterprise assets	SCAP V2

Diagram (will update to higher level)



Architecture Description

Communications (Integration Service)

Central to the OCA architecture is the common communications layer. This layer enables security products and their components to exchange security data and responses, moving away from the current point to point communications.

This communications layer uses the publish and subscribe model, enabling security components to be provided via various product packaging models but removing the need to know which product is providing which service (especially important today where capabilities like SOAR, threat intelligence, analytics and automation are delivered in various products and are constantly being moved to other packages).

Normalization Layer

Security data collection is a foundational element for security solutions. In today's environment, each product that needs this information builds its own data model, APIs, CLIs and user interface to collect this key data. This leaves the following issues:

- A single resource, like an endpoint, receives unique queries from multiple security products and must support their APIs and data model. This results in mapping the same data multiple times to feed a single product.
- Multiple connections are required (one per inquiring product) in order to pull the security data from its source. This requires configuration and maintenance for all these connections (a continuous maintenance cost).

- Security administrators will be presented with multiple, independent viewpoints (per product) on the same source information but in inconsistent presentations (data models are different).

Structured Threat Information eXpression (STIX™) is an open source language and serialization format that can help organizations collaborate and communicate more effectively. Organizations can use STIX to exchange cyber threat intelligence (CTI). CTI is represented with objects and descriptive relationships and stored as JSON for machine readability. This format delivers a consistent and machine-readable way to enable collaborative threat analysis, automated threat exchange, automated detection and response, and more.

To take advantage of STIX, you need to use STIX-shifter, an open source python library allowing software to connect to products that house data repositories by using STIX Patterning, and return results as STIX Observations.

Ontology Layer (OpenDXL Ontology)

OpenDXL enables developers join an adaptive system of interconnected services that communicate and share information to make real-time, accurate security decisions. OpenDXL provides the following benefits:

- Real time data exchange; enabling automation and orchestration across multiple vendor security defenses
- Multiple messaging patterns
  - One to many communication

    - One-one communication
- Abstraction layer protects enables communications across products via a single API across platforms, languages & protocols

---

Continuous Monitoring Layer (SCAP V2)

The **Security Content Automation Protocol (SCAP)** is a framework of key components, loosely connected via standardized interfaces, for gathering information about enterprise assets.

The primary use case for SCAP is to support automated enterprise security assessment. There are several security functions that are constantly in need of monitoring IT resources, such as:

- Vulnerability management
- Configuration management
- Software inventory and patch management

Obtaining this security information today is done product by product, polling the resource constantly for updates. SCAP defines a framework of key components, loosely connected via standardized interfaces, for gathering information about enterprise assets, storing them into a common repository for retrieval.

The SCAP components gather the data from the enterprise assets, providing a single requester for this data. SCAP focuses on supporting the diverse set of IT resources and normalizes the data collection for requesting products. SCAP provides a continuous monitoring capability for these resources. With SCAP monitoring enterprise assets , including cloud, mobile, IoT, assets, it provides a single collector for requesting products to use. This approach simplifies the data collection, removes multiple independent product polling requests (with their need for unique configuration, maintenance and performance overhead).